IBM® DB2 Universal Database™

管理ガイド: インプリメンテーション

バージョン 8.2

SC88-9133-01
(英文原典：SC09-4820-01)

IBM® DB2 Universal Database™

管理ガイド: インプリメンテーション

バージョン 8.2

SC88-9133-01
(英文原典：SC09-4820-01)

ご注意!

本書および本書で紹介する製品をご使用になる前に、『*特記事項*』に記載されている情報をお読みください。

本マニュアルに関するご意見やご感想は、次の URL からお送りください。今後の参考にさせていただきます。

`http://www.ibm.com/jp/manuals/main/mail.html`

なお、日本 IBM 発行のマニュアルはインターネット経由でもご購入いただけます。詳しくは

`http://www.ibm.com/jp/manuals/` の「ご注文について」をご覧ください。

(URL は、変更になる場合があります)

お客様の環境によっては、資料中の円記号がバックスラッシュと表示されたり、バックスラッシュが円記号と表示されたりする場合があります。

原　典： SC09-4820-01
IBM® DB2 Universal Database™
Administration Guide: Implementation
Version 8.2

発　行： 日本アイ・ビー・エム株式会社

担　当： ナショナル・ランゲージ・サポート

第1刷 2004.8

この文書では、平成明朝体™W3、平成明朝体™W9、平成角ゴシック体™W3、平成角ゴシック体™W5、および平成角ゴシック体™W7を使用しています。この(書体*)は、(財）日本規格協会と使用契約を締結し使用しているものです。フォントとして無断複製することは禁止されています。

注* 平成明朝体™W3、平成明朝体™W9、平成角ゴシック体™W3、
平成角ゴシック体™W5、平成角ゴシック体™W7

目次

本書について

3 巻で構成される本書には、 DB2 リレーショナル・データベース管理システム (RDBMS) 製品を使用し管理するために必要な、以下の情報が収められています。

- データベース設計についての情報 (*管理ガイド: プランニング*)
- データベースの使用および管理についての情報 (*管理ガイド: インプリメンテーション*)
- パフォーマンスを向上させるための、データベース環境の構成およびチューニングについての情報 (*管理ガイド: パフォーマンス*)

本書に記載されているタスクの多くは、以下に示すさまざまなインターフェースを使用して実行することができます。

- **コマンド行プロセッサー**。グラフィカル・インターフェースから、データベースをアクセスし操作することができます。このインターフェースから、SQL ステートメントおよび DB2 ユーティリティー関数も実行することができます。本書にある例のほとんどが、このインターフェースを使った場合を例として取り上げています。コマンド行プロセッサーの使用に関する詳細は、「コマンド・リファレンス」を参照してください。
- **アプリケーション・プログラミング・インターフェース**。アプリケーション・プログラム内で DB2 ユーティリティー関数を実行することができます。アプリケーション・プログラミング・インターフェースの使用についての詳細は、「*管理 API リファレンス*」を参照してください。
- **コントロール・センター**。グラフィカル・ユーザー・インターフェースを使用して、システムの構成、ディレクトリーの管理、システムのバックアップとリカバリー、ジョブのスケジューリング、およびメディアの管理などの管理タスクを実行することができます。またコントロール・センターには、システム間のデータの複製をセットアップするためのレプリケーション管理が含まれています。さらに、コントロール・センターでは、グラフィカル・ユーザー・インターフェースを介して DB2 ユーティリティー機能を実行できます。ご使用のプラットフォームによっては、コントロール・センターを呼び出すさまざまな方法があります。たとえば、 Windows プラットフォームでは、コマンド行で `db2cc` コマンドを使用するか、 DB2 フォルダーからコントロール・センター・アイコンを選択したり、あるいは「スタート」メニューを使用します。紹介のヘルプを表示するには、「コントロール・センター (Control Center)」ウィンドウの**「ヘルプ (Help)」**プルダウンから**「入門 (Getting started)」**を選択してください。 **Visual Explain** ツールは、コントロール・センターから呼び出します。

  コントロール・センターでは、3 つのビューが使用可能です。

  – 基本。このビューには、データベース、表、およびストアード・プロシージャーなどの、中核をなす DB2 UDB 機能が表示されます。
  – 詳細。このビューには、使用可能なすべてのオブジェクトおよびアクションが表示されます。エンタープライズ環境で作業しており、DB2 for z/OS または IMS に接続する場合には、このビューを使用してください。

– カスタム。このビューでは、オブジェクト・ツリーやオブジェクト・アクションを調整できるようになっています。

他にも、管理タスクを行うために使用できるツールがあります。それらのツールには、以下のものがあります。

- コマンド・エディター。これはコマンド・センターに替わるもので、 SQL ステートメントや IMS および DB2 コマンドの生成、編集、実行、および操作に使用できます。さらに、出力結果の処理、EXPLAIN された SQL ステートメントのアクセス・プランのグラフィカル表現を表示するために使用します。
- デベロップメント・センター。ネイティブ SQL Persistent Storage Module (PSM) ストアード・プロシージャー、 iSeries バージョン 5 リリース 3 以降の Java ストアード・プロシージャー、ユーザー定義関数 (UDF)、および構造化タイプをサポートします。
- ヘルス・センターは、 DBA がパフォーマンスおよびリソース割り振りの問題を解決する上で役立つツールを提供します。
- ツール設定は、コントロール・センター、ヘルス・センター、およびレプリケーション・センターの設定値を変更します。
- ジャーナルは、自動で実行するジョブをスケジュールします。
- データウェアハウス・センターは、ウェアハウス・オブジェクトを管理します。

本書の対象読者

本書は、ローカルまたはリモート・クライアントがアクセスするデータベースを設計、使用、および維持する必要のあるデータベース管理担当者、システム管理者、セキュリティー管理者、およびシステム演算子を対象としています。 DB2 Universal Database™ (DB2 UDB) リレーショナル・データベース管理システムの管理および操作について理解しておくことが必要なプログラマーや、その他のユーザーも本書をご使用になれます。

本書の構成

本書には、以下の主な項目に関する情報が記載されています。

設計のインプリメント

- 『第 1 章 データベースを作成する前に』では、データベースとデータベース内のオブジェクトを作成する前に求められる前提条件について説明します。
- 『第 2 章 DB2 Administration Server (DAS) の作成と使用』では、DAS とは何か、およびその作成方法と使用方法について取り上げます。
- 『第 3 章 データベースの作成』では、データベースとデータベース内のオブジェクトの作成に関連したタスクについて説明します。
- 『第 4 章 表および関連する表オブジェクトの作成』では、データベース設計をインプリメントする際に、特定の特性を持つ表を作成する方法について説明します。
- 『第 5 章 データベースの変更』では、データベースとデータベース内のオブジェクトの変更やドロップに関連した前提条件とタスクについて説明します。

- 『第 6 章 表および関連する表オブジェクトの変更』では、表をドロップする方法、あるいはそれらの表に関連付けられた特定の特性を変更する方法について取り上げます。関連する表オブジェクトのドロップと変更についても説明します。

データベースのセキュリティー

- 『第 7 章 データベース・アクセスの制御』では、データベース・リソースへのアクセスを制御する方法について説明します。
- 『第 8 章 DB2 Universal Database™ (DB2 UDB) アクティビティーの監査』では、データに対する不要なアクセスまたは予期せぬアクセスを検出してモニターする方法について説明します。

付録

- 『付録 A. 命名規則への準拠』では、データベースおよびオブジェクトを命名する際に従わなければならない規則を示します。
- 『付録 B. クライアントの自動転送の使用』では、クライアント・アプリケーションの自動転送およびこのサポートを使用可能にする方法を取り上げます。
- 『付録 C. Lightweight Directory Access Protocol (LDAP) ディレクトリー・サービスの使用』では、 LDAP ディレクトリー・サービスを使用する方法について説明します。
- 『付録 D. 複数のデータベース・パーティションに対するコマンドの発行』では、コマンドをパーティション・データベース環境内のすべてのパーティションに送るための *db2_all* および *rah* シェル・スクリプトの使用法について説明します。
- 『付録 E. Windows Management Instrumentation (WMI) サポートの使用』では、WMI を使用して DB2 を管理する方法について示します。
- 『付録 F. Windows NT セキュリティーの使用』では、 DB2 が Windows セキュリティーを処理する方法について説明します。
- 『付録 G. Windows パフォーマンス・モニターの使用』では、 Windows パフォーマンス・モニターを使用して、 DB2 パフォーマンス・データを収集する方法について説明します。
- 『付録 H. Windows データベース・パーティション・サーバーの使用』では、Windows でパーティション・データベースを処理するために使用されるユーティリティーについて説明します。
- 『付録 I. 複数の論理データベース・パーティションの構成』では、パーティション・データベース環境で複数論理データベース・パーティションを構成する方法について説明します。
- 『付録 J. コントロール・センターの拡張』では、新しいアクションを割り当てた新しいツールバー・ボタンの追加、新しいオブジェクト定義の追加、および新しいアクション定義の追加により、コントロール・センターを拡張する方法について説明します。

「*管理ガイド: インプリメンテーション*」から、 『Utilities for Moving Data』という章が移されました。

注: 「コマンド・リファレンス」および「*管理 API リファレンス*」に含まれていた、データの移動に使用する DB2 ユーティリティーに関するすべての情報、およびそれに類似したトピックが、「*データ移動ユーティリティー ガイドおよびリファレンス*」に統合されました。

「*データ移動ユーティリティー ガイドおよびリファレンス*」が、これらのトピックに関する主要かつ唯一の情報源です。

データの複製に関する詳細については、「*IBM DB2 Information Integrator SQL レプリケーション・ガイドおよびリファレンス*」を参照してください。

「*管理ガイド: インプリメンテーション*」から、 『Recovering a Database』という章が移されました。

注: 「コマンド・リファレンス」および「*管理 API リファレンス*」に含まれていた、データのバックアップおよびリカバリーの方法およびツールに関するすべての情報が、「*データ・リカバリーと高可用性 ガイドおよびリファレンス*」に統合されました。

「*データ・リカバリーと高可用性 ガイドおよびリファレンス*」が、これらのトピックに関する主要かつ唯一の情報源です。

他の管理ガイドの巻の概要

管理ガイド : プランニング

「*管理ガイド: プランニング*」は、データベース設計を扱っています。論理設計および物理設計、分散トランザクションについて説明しています。この巻のそれぞれの章と付録は、以下のように構成されています。

データベースの概念

- 『リレーショナル・データベースの基本的な概念』では、リカバリー・オブジェクト、ストレージ・オブジェクト、およびシステム・オブジェクトを含むデータベース・オブジェクトの概要を説明します。
- 『並列データベース・システム』では、 DB2 で実現される並列処理のタイプを紹介します。
- 『データウェアハウジングについて』では、データウェアハウジングおよびウェアハウジング・タスクについて概説します。

データベースの設計

- 『論理データベースの設計』では、論理データベースの設計に関する概念と指針について説明します。
- 『物理データベースの設計』では、物理データベースの設計 (データ・ストレージに関する考慮事項を含む) の指針について説明します。
- 『分散データベースの設計』では、単一トランザクションで複数のデータベースをアクセスする方法を説明します。
- 『トランザクション・マネージャーの設計』では、分散トランザクション処理環境でデータベースを使用する方法について説明します。

付録

- 『リリース間の非互換性』では、バージョン 7 とバージョン 8 での非互換性と、意識していなければならない将来の非互換性について示します。
- 『各国語サポート (NLS)』では、テリトリー、言語、およびコード・ページの情報を含む、 DB2 各国語サポート (NLS) について説明します。
- 『64 ビット環境におけるラージ・ページのサポートの使用可能化 (AIX)』では、16 MB ページ・サイズのサポート、およびこのサポートを使用可能にする方法を取り上げます。

管理ガイド : パフォーマンス

「*管理ガイド: パフォーマンス*」では、パフォーマンスに関する問題、つまり、アプリケーションや DB2 Universal Database 製品のパフォーマンスの設定、テスト、および改善に関連したトピックや問題を扱います。この巻のそれぞれの章と付録は、以下のように構成されています。

パフォーマンスの紹介

- 『パフォーマンスの紹介』では、 DB2 UDB パフォーマンスの管理と改善に関する概念と考慮事項について紹介します。
- 『アーキテクチャーとプロセス』では、基礎となる DB2 Universal Database の構造およびプロセスを紹介します。

アプリケーション・パフォーマンスのチューニング

- 『アプリケーションについての考慮事項』では、アプリケーションの設計時点で、データベースのパフォーマンスを向上させる手法をいくつか説明します。
- 『環境についての考慮事項』では、データベース環境の設定時点で、データベースのパフォーマンスを向上させる手法をいくつか説明します。
- 『システム・カタログ統計』では、最適なパフォーマンスを達成するために、データについての統計を収集し使用する方法について説明します。
- 『SQL コンパイラーに関する解説』では、 SQL コンパイラーを使用してコンパイルしたときに、 SQL ステートメントがどうなるかについて説明します。
- 『SQL Explain 機能』では、Explain 機能について説明します。この機能により、データにアクセスするために SQL コンパイラーが行った選択を調べることができます。

システムのチューニングと構成

- 『操作上のパフォーマンス』では、データベース・マネージャーがメモリーを使用する方法の概要、およびランタイムのパフォーマンスに影響を与えるその他の考慮事項について説明します。
- 『管理プログラムの使用法』では、データベース管理のある局面を制御するための管理プログラムの使用について紹介します。
- 『構成のスケーリング』では、データベース・システムのサイズの増加に関連する考慮事項と作業について説明します。
- 『データベース・パーティション間でのデータの再分散』では、パーティション間でデータを再配分するために、パーティション・データベース環境において必要な作業について説明します。

- 『ベンチマーク・テスト』では、ベンチマーク・テストの概要とベンチマーク・テストの実行方法について説明します。
- 『DB2 の構成』では、データベース・マネージャー、データベース構成ファイルと、データベース・マネージャー、データベース、および DAS 構成パラメーターの値について説明します。

付録

- 『DB2 登録変数と環境変数』では、プロファイル・レジストリーの値と環境変数を示します。
- 『Explain 表と定義』では、 DB2 Explain 機能が使用する表と、それらの表の作成方法について説明します。
- 『SQL Explain ツール』では、 DB2 Explain ツールである db2expln および dynexpln の使用法について説明します。
- 『db2exfmt － Explain 表フォーマット・ツール』では、 DB2 Explain ツールを使用して Explain 表データをフォーマットする方法について説明します。

第 1 部 設計のインプリメント

第 1 章 データベースを作成する前に

データベースの設計が決定したら、データベースとその中のオブジェクトを作成しなければなりません。オブジェクトには、スキーマ、データベース・パーティション・グループ、表スペース、表、ビュー、ラッパー、 サーバー、ニックネーム、タイプのマッピング、関数のマッピング、別名、 ユーザー定義タイプ (UDT)、ユーザー定義関数 (UDF)、自動サマリー表 (AST)、トリガー、制約、索引、 およびパッケージがあります。これらのオブジェクトは、コマンド行プロセッサーまたはアプリケーションで SQL ステートメントを使って作成できます。

SQL ステートメントの詳細については、「*SQL リファレンス*」を参照してください。コマンド行プロセッサーの詳細については、「*コマンド・リファレンス*」を参照してください。アプリケーション・プログラミング・インターフェース (API) について、詳しくは「*管理 API リファレンス*」を参照してください。

データベース・オブジェクトを作成するもう 1 つの方法は、コントロール・センターです。コントロール・センターは、SQL ステートメント、コマンド行プロセッサー・コマンド、または API の代わりに使用することができます。

この章では、コントロール・センターを使用してタスクを完了する方法を、囲み線の中に表記して強調しています。同じことをコマンド行から行う方法を、そのすぐ後ろに示します。例が併記されている場合もあります。また、一方の方法しか示されていないタスクもあります。コントロール・センターで作業を行う際には、ヘルプを使用して、この章に記載されている概要よりも詳しい情報を参照できます。

この章では、データベースとそのすべてのオブジェクトを作成する前に知っている必要がある情報を中心に説明します。前提条件となる概念とトピック、およびデータベースの作成前に済ませておくべきタスクを示します。

次の章には、データベース設計のインプリメンテーションを構成する、さまざまなオブジェクトに関する短い説明があります。

この部の最後の章には、データベースを変更する前に考慮しなければならないトピックと、データベース・オブジェクトの変更方法やドロップ方法について説明されています。

この章やその後の章で説明するトピックの一部は、 DB2 Universal Database がオペレーティング・システムと対話を行う部分で、オペレーティング・システム固有の違いがある場合があります。 DB2 UDB によって提供されるものではなく、ネイティブのオペレーティング・システムの機能および違いを利用することができます。相違点を正確に知るには、「*概説およびインストール*」、およびオペレーティング・システムの資料を参照してください。

1 つの例として、Windows は、「サービス」と呼ばれるアプリケーション・タイプをサポートします。 DB2 for Windows には、サービスとして定義されたそれぞれの DB2 インスタンスがあります。サービスはシステム・ブート時に自動的に開始できますが、これは、サービス制御パネル・アプレットを通してユーザーによって

行われるか、または Microsoft 32 ビット・アプリケーション・プログラミング・インターフェース (API) に含まれるサービス機能を使用する Microsoft 32 ビット・アプリケーションによって行われます。サービスは、システムにログオンしているユーザーがなくても実行できます。

特に明記しない場合、Windows 9x とは Windows 98 および Windows ME のことです。 Windows NT とは、Windows NT のほかに Windows 2000、 Windows XP、および Windows Server 2003 を指します。単に Windows という場合は、サポートされるすべての Windows オペレーティング・システムを指します。

インスタンスを使った作業

データベースを実現する前に、以下の前提条件タスクについて理解している必要があります。

- 『UNIX での DB2 UDB の開始』
- 5 ページの『Windows での DB2 UDB の開始』
- 6 ページの『データベース・マネージャーの複数インスタンス』
- 8 ページの『スキーマ別のオブジェクトのグループ化』
- 8 ページの『並列処理』
- 13 ページの『データベースでのデータ・パーティションを使用可能にする』
- 14 ページの『UNIX でのインスタンスの停止』

UNIX での DB2 UDB の開始

通常の業務を行っている最中に DB2 Universal Database™ (DB2 UDB) を開始または停止する必要がある場合があります。たとえば、以下のタスクを実行する前に、インスタンスを開始しなければなりません。

- インスタンスのデータベースに接続する。
- アプリケーションをプリコンパイルする。
- データベースにパッケージをバインドする。
- ホスト・データベースにアクセスする。

前提条件:

システムで DB2 UDB インスタンスを開始するには、次のようにします。

1. インスタンスに対する SYSADM、SYSCTRL、または SYSMAINT 権限を持つユーザー ID またはユーザー名を使ってログインします。あるいは、インスタンス所有者としてログインします。
2. 以下のように、始動スクリプトを実行します。

```
. INSTHOME/sqllib/db2profile    (Bourne または Korn シェルの場合)
source INSTHOME/sqllib/db2cshrc (C シェルの場合)
```

INSTHOME は、使用するインスタンスのホーム・ディレクトリーです。

手順:

インスタンスを開始するには、以下の 2 つの方法のいずれかを使用します。

1. コントロール・センターを使用してインスタンスを開始するには、以下のようにします。

   1. オブジェクト・ツリーを順に展開し、**「インスタンス (Instances)」**フォルダーを表示します。
   2. 開始するインスタンスを右クリックして、ポップアップ・メニューから**「開始 (start)」**を選択します。

2. コマンド行を使用してインスタンスを開始するには、以下のように入力します。

   ```
   db2start
   ```

関連タスク:

- 14 ページの『UNIX でのインスタンスの停止』
- 27 ページの『現行インスタンスの設定』
- 5 ページの『Windows での DB2 UDB の開始』

Windows での DB2 UDB の開始

通常の業務を行っている最中に DB2 Universal Database™ (DB2 UDB) を開始または停止する必要がある場合があります。たとえば、以下のタスクを実行する前に、インスタンスを開始しなければなりません。

- インスタンスのデータベースに接続する。
- アプリケーションをプリコンパイルする。
- データベースにパッケージをバインドする。
- ホスト・データベースにアクセスする。

前提条件:

db2start から DB2 UDB をサービスとして正常に立ち上げるには、そのユーザー・アカウントで Windows サービスを開始するために Windows NT オペレーティング・システムで定義された、適切な特権が必要です。ユーザー・アカウントは、管理者、サーバー・オペレーター、またはパワー・ユーザーのいずれかのグループのメンバーです。

手順:

インスタンスを開始するには、以下の 2 つの方法のいずれかを使用します。

1. コントロール・センターを使用してインスタンスを開始するには、以下のようにします。

   1. オブジェクト・ツリーを順に展開し、**「インスタンス (Instances)」**フォルダーを表示します。
   2. 開始するインスタンスを右クリックして、ポップアップ・メニューから**「開始 (start)」**を選択します。

2. コマンド行を使用してインスタンスを開始するには、以下のように入力します。

   ```
   db2start
   ```

db2start コマンドを使用すると、DB2 UDB は Windows サービスとして立ち上がります。 **db2start** を呼び出すときにスイッチ ″/D″ を指定すると、 Windows での

DB2 UDB をプロセスとして実行することができます。「コントロール パネル」または ″NET START″ コマンドを使用して、 DB2 UDB をサービスとして開始することもできます。

パーティション・データベース環境で実行しているときには、各データベース・パーティション・サーバーは Windows サービスとして開始されます。パーティション・データベース環境で DB2 をプロセスとして開始するために、「/D」スイッチを使用することはできません。

関連タスク:

- 4 ページの『UNIX での DB2 UDB の開始』
- 14 ページの『UNIX でのインスタンスの停止』
- 15 ページの『Windows でのインスタンスの停止』

データベース・マネージャーの複数インスタンス

1 つのサーバーにデータベース・マネージャーの複数インスタンスを作成することができます。これはつまり、物理的に 1 つのマシンに同じ製品のインスタンスを複数作成し、それらを並行して稼働させることができるということです。これにより、複数の環境を柔軟に設定できます。

次の環境を作成するために、複数のインスタンスが必要になる場合があります。

- 開発環境を実稼働環境から分離する。
- インスタンスがサービスする特定のアプリケーションのそれぞれを別個に調整する。
- 機密情報を管理担当者から保護する。たとえば、給与計算データベースをそのインスタンスで保護し、他のインスタンスの所有者が給与計算データを見ることができないようにすることができます。

注: (UNIX® オペレーティング・システムのみ) 複数のインスタンス間での環境競合を防ぐために、各インスタンスごとにホーム・ファイル・システムを設定する必要があります。ホーム・ファイル・システムが共有されている場合、エラーが戻されます。

DB2® Universal Database (DB2 UDB) プログラム・ファイルは、物理的には特定のマシンの 1 つのロケーションに保管されます。作成される各インスタンスは、このロケーションを指しているので、作成されるインスタンスごとにプログラム・ファイルが複写されるわけではありません。いくつかの関連するデータベースを、単一のインスタンス内に置くことができます。

インスタンスはノード・ディレクトリーにローカルまたはリモートのいずれかとしてカタログされます。デフォルト・インスタンスは DB2INSTANCE 環境変数で定義されます。データベースの作成、アプリケーションの強制終了、データベースのモニター、またはデータベース・マネージャー構成の更新などの、インスタンス・レベルでしか行うことのできない保守およびユーティリティー・タスクを実行するために、他のインスタンスに **ATTACH** (アタッチ) することができます。デフォルト・インスタンスにないインスタンスにアタッチしようとすると、そのインスタンスとの通信方法を判別するためにノード・ディレクトリーが使用されます。

関連概念:

- 21 ページの『UNIX オペレーティング・システムでの複数インスタンス』
- 22 ページの『Windows オペレーティング・システムでの複数インスタンス』

関連タスク:

- 23 ページの『追加のインスタンスの作成』

関連資料:

- 「コマンド・リファレンス」の『ATTACH コマンド』

データベース・マネージャーの別のインスタンスへのアタッチ

(リモート・インスタンスを含む) 別のインスタンスにアタッチするには、 **ATTACH** コマンドを使用します。

前提条件:

1 つまたは複数のインスタンスがすでに存在しなければなりません。

手順:

コントロール・センターを使ってデータベース・マネージャーの別のインスタンスにアタッチするには、以下のようにします。

1. オブジェクト・ツリーを順に展開し、**「インスタンス (Instances)」**フォルダーを表示します。
2. アタッチしたいインスタンスをクリックします。
3. 選択したインスタンス名を右クリックします。
4. 「アタッチ - DB2 (Attach-DB2)」ウィンドウで、ユーザー ID とパスワードを入力し、**「OK」**をクリックします。

コマンド行を使用してインスタンスにアタッチするには、以下のように入力します。

```
db2 attach to <instance name>
```

たとえば、以下のようにコマンドを出すと、以前にノード・ディレクトリーにカタログされた testdb2 というインスタンスにアタッチします。

```
db2 attach to testdb2
```

testdb2 インスタンスの保守に関連した作業を実行した後で、次のコマンドを実行すると、そのインスタンスから **DETACH** することができます。

```
db2 detach
```

関連資料:

- 「コマンド・リファレンス」の『ATTACH コマンド』
- 「コマンド・リファレンス」の『DETACH コマンド』

スキーマ別のオブジェクトのグループ化

データベース・オブジェクトは、1 つの ID で構成されているか、 2 つの ID で構成されるスキーマ*修飾オブジェクト* です。スキーマ修飾オブジェクトのスキーマまたは高位部分は、データベースの中のオブジェクトを分類またはグループ化するための手段を提供します。表、ビュー、別名、特殊タイプ、関数、索引、パッケージ、またはトリガーが作成されると、それがスキーマに割り当てられます。この割り当ては、明示的または暗黙的のいずれかで行われます。

ステートメント中のオブジェクトを参照するときに 2 部から成るオブジェクト名の高位部分を使用する場合は、スキーマを明示的に使用することになります。たとえば、USER A がスキーマ C の CREATE TABLE ステートメントを次のように発行します。

```
CREATE TABLE C.X (COL1 INT)
```

2 部から成るオブジェクト名の高位部分を使用しない場合は、スキーマを暗黙的に使用することになります。スキーマを暗黙的に使用すると、オブジェクト名の高位部分を構成するのに使用されるスキーマ名を識別するために、 CURRENT SCHEMA 特殊レジスターが使用されます。 CURRENT SCHEMA の初期値は、現行セッション・ユーザーの許可 ID です。これを現行セッション中に変更したい場合は、 SET SCHEMA ステートメントを使用して特殊レジスターを別のスキーマ名に設定することができます。

データベース作成時、一部のオブジェクトは、特定のスキーマ内に作成されてシステム・カタログ表に保管されます。

動的 SQL ステートメントでは、スキーマ修飾オブジェクト名は、修飾なしオブジェクト名参照の修飾子として CURRENT SCHEMA 特殊レジスター値を暗黙的に使用します。静的 SQL ステートメントでは、QUALIFIER プリコンパイル/ BIND オプションは、修飾なしデータベース・オブジェクト名の修飾子を暗黙的に指定します。

独自のオブジェクトを作成する前に、それらをデフォルトのスキーマの中に作成するか、または論理的にオブジェクトをグループ化する別のスキーマを使用するかを検討しなければなりません。共用されるオブジェクトを作成している場合は、別のスキーマ名を使用すると、非常に便利です。

関連概念:

- 75 ページの『システム・カタログ表の定義』

関連タスク:

- 94 ページの『スキーマの作成』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『SET SCHEMA ステートメント』
- 「*SQL リファレンス 第 1 巻*」の『CURRENT SCHEMA 特殊レジスター』

並列処理

データベース・パーティション内または非パーティション・データベース内で並列処理を利用するためには、構成パラメーターを修正しなければなりません。たとえ

ば、パーティション内並列処理を使用して、対称マルチプロセッサー (SMP) マシン上の複数のプロセッサーを利用することができます。

照会のパーティション間並列処理を使用可能にする

手順:

パーティション間並列処理は、データベース・パーティションの数およびこれらのパーティションにわたるデータの分散に基づいて、自動的に行われます。

関連概念:

- 「*管理ガイド: プランニング*」の『パーティションおよびプロセッサー環境』
- 「*管理ガイド: プランニング*」の『データのパーティション』
- 「*管理ガイド: プランニング*」の『データベース・パーティション・グループの設計』
- 「*管理ガイド: パフォーマンス*」の『パーティション・データベースのパーティション』

関連タスク:

- 9 ページの『照会のパーティション内並列処理を使用可能にする』
- 13 ページの『データベースでのデータ・パーティションを使用可能にする』
- 「*管理ガイド: パフォーマンス*」の『パーティション間でのデータの再分散』

照会のパーティション内並列処理を使用可能にする

手順:

コントロール・センターを使用して、特定のデータベースまたはデータベース・マネージャー構成ファイルの中の個々の項目の値を調べたり修正したりできます。

このほか **GET DATABASE CONFIGURATION** コマンドおよび **GET DATABASE MANAGER CONFIGURATION** コマンドを使用して、特定のデータベースまたはデータベース・マネージャー構成ファイルの中の個々の項目の値を調べることもできます。特定のデータベースまたはデータベース・マネージャー構成ファイルの個々の項目を修正するためには、それぞれ **UPDATE DATABASE CONFIGURATION** コマンドと **UPDATE DATABASE MANAGER CONFIGURATION** コマンドを使用します。

パーティション内並列処理に影響を与える構成パラメーターには、 *max_querydegree* と *intra_parallel* のデータベース・マネージャー・パラメーター、および *dft_degree* データベース・パラメーターがあります。

照会のパーティション内並列処理を行わせるためには、 1 つまたは複数のデータベース構成パラメーター、データベース・マネージャー構成パラメーター、プリコンパイル・オプションまたは BIND オプション、または特殊レジスターを修正する必要があります。

intra_parallel
このデータベース・マネージャー構成パラメーターは、データベース・マネ

ージャーでパーティション内並列処理を使用できるかどうかを指定します。デフォルトでは、パーティション内並列処理を使用しません。

max_querydegree
このデータベース・マネージャー構成パラメーターは、このインスタンスで実行中の SQL ステートメントで使用される、パーティション内並列処理の最大度を指定します。 SQL ステートメントは、パーティション内で並列して操作を行うとき、このパラメーターを超える数を使用することはありません。さらに、*max_querydegree* の値を使用するためには、構成パラメーター *intra_parallel* を ″YES″ に設定する必要もあります。この構成パラメーターのデフォルト値は -1 です。この値は、オプティマイザーによって決められた並列処理の度合いがシステムで使用されることを意味します。それ以外の場合は、ユーザー指定の値が使用されます。

dft_degree
データベース構成パラメーター。 DEGREE BIND オプションおよび CURRENT DEGREE 特殊レジスターに対するデフォルトを提供します。デフォルト値は 1 です。値 ANY は、オプティマイザーによって決められた並列処理の度合いがシステムで使用されることを意味します。

DEGREE
静的 SQL に対するプリコンパイルまたはバインドのオプション。

CURRENT DEGREE
動的 SQL に対する特殊レジスター。

関連概念:

- 「*管理ガイド: パフォーマンス*」の『アプリケーションの並列処理』
- 「*管理ガイド: パフォーマンス*」の『並列処理の情報』

関連タスク:

- 「*管理ガイド: パフォーマンス*」の『構成パラメーターによる DB2 の構成』

関連資料:

- 「*管理ガイド: パフォーマンス*」の『max_querydegree - 「照会の最大並列処理の度合い」構成パラメーター』
- 「*管理ガイド: パフォーマンス*」の『intra_parallel - 「パーティション内並列処理の使用可能化」構成パラメーター』
- 「*管理ガイド: パフォーマンス*」の『dft_degree - 「デフォルトの並列処理の度合い」構成パラメーター』
- 「*コマンド・リファレンス*」の『BIND コマンド』
- 「*コマンド・リファレンス*」の『PRECOMPILE コマンド』
- 「*SQL リファレンス 第 1 巻*」の『CURRENT DEGREE 特殊レジスター』

ユーティリティーのパーティション内並列処理を使用可能にする

この節では、以下のユーティリティーについて、パーティション内並列処理を可能にする方法の概要について説明します。

- ロード
- 索引作成
- データベースまたは表スペースのバックアップ

- データベースまたは表スペースのリストア

ユーティリティーのパーティション間並列処理は、データベース・パーティションの数に基づいて自動的に行われます。

データのロードの並列処理を使用可能にする: ロード・ユーティリティーは、自動的に並列処理を使用可能にします。または、**LOAD** コマンドで以下のパラメーターを使うことができます。

- CPU_PARALLELISM
- DISK_PARALLELISM

パーティション・データベース環境では、ターゲット表が複数のパーティション上に定義されるとき、データ・ロード用のパーティション間並列処理が自動的に発生します。データ・ロード用のパーティション間並列処理は、OUTPUT_DBPARTNUMBS を指定することによってオーバーライドできます。さらにロード・ユーティリティーもまた、ターゲット・パーティションのサイズに応じてデータ・パーティション並列処理を自動的に使用可能にします。 Load ユーティリティーによって選択される並列処理の度合いの最大値を制御するには、MAX_NUM_PART_AGENTS を使用することができます。データ・パーティション並列処理は、 ANYORDER とともに PARTITIONING_DBPARTNUMS を指定することによってオーバーライドできます。

関連概念:

- 「*データ移動ユーティリティー ガイドおよびリファレンス*」の『ロードの概要』
- 「*データ移動ユーティリティー ガイドおよびリファレンス*」の『パーティション・データベース・ロードの概説』

索引の作成の並列処理を使用可能にする: 索引の作成中に並列処理を使用可能にするには、以下のとおりにしてください。

- *intra_parallel* データベース・マネージャー構成パラメーターは ON でなければなりません。
- 表は、並列処理の益が得られる十分な大きさでなければなりません。
- 1 つの SMP マシンで、複数プロセッサーが使用可能でなければなりません。

関連資料:

- 「*管理ガイド: パフォーマンス*」の『intra_parallel - 「パーティション内並列処理の使用可能化」構成パラメーター』
- 「*SQL リファレンス 第 2 巻*」の『CREATE INDEX ステートメント』

データベースまたは表スペースのバックアップで入出力並列処理を使用可能にする: データベースまたは表スペースのバックアップで入出力並列処理を使用可能にするには、以下のようにします。

- 複数の宛先メディアを使用します。
- 複数のコンテナーを定義することによって並列入出力用の表スペースを構成します。または、複数ディスクを使用する単一のコンテナーを使い、DB2_PARALLEL_IO レジストリー変数を適切に設定します。並列入出力を利用したい場合は、コンテナーを定義する前に必要な作業の意味を考慮してください。

これは、必要が生じるたびに常に行えるとは限りません。データベースや表スペースをバックアップする必要が生じる前に、あらかじめ計画しておく必要があります。

- **BACKUP** コマンドで PARALLELISM パラメーターを使用し、並列処理の度合いを指定します。
- **BACKUP** コマンドで WITH num-buffers BUFFERS パラメーターを使用し、並列処理の度合いに見合う十分なバッファーを使用できるようにします。バッファーの数は、宛先メディア、選択した並列処理の度合い、そして多少の余分を合計した数に等しいものにします。

  また以下のような、バックアップ・バッファー・サイズを使用します。

  – 可能な限り大きなサイズにする。 4 MB か 8 MB (1024 か 2048ページ) が良いようです。
  – 少なくとも、バックアップする表スペースの可能な最大数を含められるだけの大きさにする (extentsize * コンテナー数)。

関連資料:

- 「コマンド・リファレンス」の『BACKUP DATABASE コマンド』

データベースまたは表スペースのリストアで入出力並列処理を使用可能にする: データベースまたは表スペースのリストアで入出力並列処理を使用可能にするには、以下のようにします。

- 複数のソース・メディアを使用します。
- 並列入出力用に表スペースを構成します。コンテナーを定義する前に、このオプションの使用について決定しておく必要があります。これは、必要が生じるたびに常に行えるとは限りません。データベースや表スペースをリストアする必要が生じる前に、あらかじめ計画しておく必要があります。
- **RESTORE** コマンドで PARALLELISM パラメーターを使用し、並列処理の度合いを指定します。
- **RESTORE** コマンドで WITH num-buffers BUFFERS パラメーターを使用し、並列処理の度合いに見合う十分なバッファーを使用できるようにします。バッファーの数は、宛先メディア、選択した並列処理の度合い、そして多少の余分を合計した数に等しいものにします。

  また以下のような、リストア・バッファー・サイズを使用します。

  – 可能な限り大きなサイズにする。 4 MB か 8 MB (1024 か 2048ページ) が良いようです。
  – 少なくとも、リストアする表スペースの可能な最大数を含められるだけの大きさにする (extentsize * コンテナー数)。
  – バックアップ・バッファー・サイズと同じか、その倍数である。

関連資料:

- 「コマンド・リファレンス」の『RESTORE DATABASE コマンド』

データベースでのデータ・パーティションを使用可能にする

データベースをパーティション化環境で使用するかどうかの決定は、データベース作成前に行う必要があります。データベース設計の一部として、データベース・パーティションによるパフォーマンス改善を利用するかどうかを決定しておかなければなりません。

パーティション・データベースの作成に関連した考慮事項が、以下に示されます。

手順:

パーティション・データベース環境で実行している場合、 **CREATE DATABASE** コマンドまたは sqlecrea() アプリケーション・プログラミング・インターフェース (API) を使用して、 db2nodes.cfg ファイル内に存在するどのノードからでもデータベースを作成することができます。

パーティション・データベース作成前に、どのデータベース・パーティションをそのデータベースのカタログ・ノードとするかを選択しなければなりません。その後、そのパーティションからデータベースを直接作成するか、またはそのパーティションにアタッチされたリモート・クライアントからデータベースを作成できます。アタッチして CREATE DATABASE コマンドを実行するデータベース・パーティションは、その特定のデータベースに対する*カタログ・ノード* になります。

カタログ・ノードは、すべてのシステム・カタログ表が保管されるデータベース・パーティションです。システム表に対するすべてのアクセスは、このデータベース・パーティションを通して行わなければなりません。フェデレーテッド・データベース・オブジェクト (ラッパー、サーバー、ニックネームなど) はすべて、このノードのシステム・カタログ表に保管されます。

可能であれば、各データベースを別個のインスタンスの中に作成してください。これが可能でない場合 (つまり、1 インスタンス当たり複数のデータベースを作成しなければならない場合)、カタログ・ノードを使用可能なデータベース・パーティションに分散させる必要があります。これを行うと、単一データベース・パーティションにおけるカタログ情報の競合が削減されます。

注: 他のデータでバックアップに必要な時間が増えてしまうため、定期的にカタログ・ノードのバックアップをとり、 (可能ならば) そこにユーザー・データを書き込むのを避けるべきです。

データベースを作成すると、 db2nodes.cfg ファイルに定義されたすべてのデータベース・パーティションにわたって自動的に作成されます。

システムに最初のデータベースが作成されると、システム・データベース・ディレクトリーが作成されます。これは、作成した他のデータベースについての情報と一緒に追加されます。 UNIX の場合、システム・データベース・ディレクトリーは sqldbdir であり、 ホーム・ディレクトリーの下、または DB2 Universal Database™ (DB2 UDB) インストール・ディレクトリーの下の sqllib ディレクトリーに配置されます。 UNIX では、このディレクトリーは、パーティション・データベースを形成するすべてのデータベース・パーティションに対する唯一のシステム・データベース・ディレクトリーであるため、共有ファイル・システム (たとえば、UNIX プ

ラットフォーム上の NFS) に常駐しなければなりません。 Windows の場合、システム・データベース・ディレクトリーはインスタンス・ディレクトリー内に置かれます。

さらに、sqldbdir ディレクトリーにはシステム・インテンション・ファイルも置かれます。これは sqldbins と呼ばれ、データベース・パーティションが同期を維持できるようにするものです。このファイルも、すべてのデータベース・パーティションにわたって 1 つのディレクトリーしかないため、共有ファイル・システムに常駐しなければなりません。このファイルは、データベースを形成するすべてのパーティションで共用されます。

データ・パーティションを利用するためには、構成パラメーターを修正しなければなりません。 **GET DATABASE CONFIGURATION** コマンドおよび **GET DATABASE MANAGER CONFIGURATION** コマンドを使用して、特定のデータベースまたはデータベース・マネージャー構成ファイルの中の個々の項目の値を調べることができます。特定のデータベースまたはデータベース・マネージャー構成ファイルの個々の項目を修正するためには、それぞれ **UPDATE DATABASE CONFIGURATION** コマンドと **UPDATE DATABASE MANAGER CONFIGURATION** コマンドを使用します。

パーティション・データベースに影響を与えるデータベース・マネージャー構成パラメーターには、 *conn_elapse*、 *fcm_num_anchors*、 *fcm_num_buffers*、 *fcm_num_connect*、 *fcm_num_rqb*、 *max_connretries*、 *max_coordagents*、 *max_time_diff*、 *num_poolagents*、および *stop_start_time* があります。

関連タスク:

- 「*管理ガイド: パフォーマンス*」の『構成パラメーターによる DB2 の構成』

関連資料:

- 「*管理 API リファレンス*」の『sqlecrea - データベースの作成』
- 「コマンド・リファレンス」の『CREATE DATABASE コマンド』

UNIX でのインスタンスの停止

データベース・マネージャーの現在のインスタンスを停止する必要があるかもしれません。

前提条件:

システムでインスタンスを停止するには、以下のことを行わなければなりません。

1. インスタンスに対する SYSADM、SYSCTRL、または SYSMAINT 権限を持つユーザー ID またはユーザー名を使ってインスタンスにログインまたはアタッチします。あるいは、インスタンス所有者としてログインします。
2. 停止したい特定のデータベースに接続された、すべてのアプリケーションおよびユーザーを表示します。重要なアプリケーションまたはクリティカルなアプリケーションが実行されていないことを確認するために、アプリケーションをリストします。リストを表示するには、SYSADM、SYSCTRL、または SYSMAINT 権限が必要です。

3. すべてのアプリケーションおよびユーザーにデータベースの使用を中断させます。ユーザーに強制するためには、SYSADM または SYSCTRL 権限が必要です。

制約事項:

db2stop コマンドはサーバーでのみ実行できます。このコマンドの実行中はデータベースの接続は許されません。接続されたインスタンスがあると、インスタンスが停止する前に強制的にオフにされます。

注: コマンド行プロセッサーのセッションがインスタンスにアタッチされたら、**terminate** コマンドを実行し、各セッションを終了してから **db2stop** コマンドを実行します。 **db2stop** コマンドを実行すると、 DB2INSTANCE 環境変数で定義されたインスタンスが停止します。

手順:

インスタンスを停止するには、以下の 2 つの方法のいずれかを使用します。

1. コントロール・センターを使用してインスタンスを停止するには、以下のようにします。

   1. オブジェクト・ツリーを順に展開し、**「インスタンス (Instances)」**フォルダーを表示します。
   2. 停止したいインスタンスを 1 つずつクリックします。
   3. 選択したすべてのインスタンスを右クリックして、ポップアップ・メニューから**「停止 (stop)」**を選択します。
   4. 「停止の確認 (Confirm stop)」ウィンドウで、**「OK」**をクリックします。

2. コマンド行を使用してインスタンスを停止するには、以下のように入力します。

   ```
   db2stop
   ```

db2stop コマンドを使用して、パーティション・データベース環境内の個々のパーティションを停止またはドロップすることができます。パーティション・データベースを使用し、以下のコマンドで論理パーティションをドロップする場合、

```
db2stop drop nodenum <0>
```

そのデータベースへのアクセスを試行しているユーザーがいないことを確認してください。そのようなユーザーが存在する場合、エラー・メッセージ SQL6030N が受信されます。

関連資料:

- 「コマンド・リファレンス」の『db2stop - DB2 の停止コマンド』
- 「コマンド・リファレンス」の『TERMINATE コマンド』

Windows でのインスタンスの停止

データベース・マネージャーの現在のインスタンスを停止する必要があるかもしれません。

前提条件:

システムでインスタンスを停止するには、以下のことを行わなければなりません。

1. DB2 Universal Database™ (DB2 UDB) サービスの停止を行うユーザー・アカウントは、 Windows オペレーティング・システムで定義された適切な特権を持っている必要があります。ユーザー・アカウントは、管理者、サーバー・オペレーター、またはパワー・ユーザーのいずれかのグループのメンバーです。
2. 停止したい特定のデータベースに接続された、すべてのアプリケーションおよびユーザーを表示します。重要なアプリケーションまたはクリティカルなアプリケーションが実行されていないことを確認するために、アプリケーションをリストします。リストを表示するには、SYSADM、SYSCTRL、または SYSMAINT 権限が必要です。
3. すべてのアプリケーションおよびユーザーにデータベースの使用を中断させます。ユーザーに強制するためには、SYSADM または SYSCTRL 権限が必要です。

制約事項:

db2stop コマンドはサーバーでのみ実行できます。このコマンドの実行中はデータベースの接続は許されません。接続されたインスタンスがあると、DB2 UDB が停止する前に強制的にオフにされます。

注: コマンド行プロセッサーのセッションがインスタンスにアタッチされたら、**terminate** コマンドを実行し、各セッションを終了してから **db2stop** コマンドを実行します。 **db2stop** コマンドを実行すると、 DB2INSTANCE 環境変数で定義されたインスタンスが停止します。

手順:

システムのインスタンスを停止するには、以下のいずれかの方法に従ってください。

- **db2stop**
- コントロール・センターを使ってサービスを停止する

1. オブジェクト・ツリーを順に展開し、**「インスタンス (Instances)」**フォルダーを表示します。
2. 停止したいインスタンスを 1 つずつクリックします。
3. 選択したすべてのインスタンスを右クリックして、ポップアップ・メニューから**「停止 (stop)」**を選択します。
4. 「停止の確認 (Confirm stop)」ウィンドウで、**「OK」**をクリックします。

- 『NET STOP』 コマンドを使って停止する。
- アプリケーション内からインスタンスを停止する。

パーティション・データベース環境で DB2 UDB を実行しているときには、各データベース・パーティション・サーバーがサービスとして開始されることに注意してください。それぞれのサービスを停止する必要があります。

関連資料:

- 「コマンド・リファレンス」の『db2stop - DB2 の停止コマンド』

データベースを作成するための準備

実際にデータベースを作成する前に、考慮すべき概念や作業がたくさんあります。これらの概念および作業には、データベースを設計して、データベースで必要とされるインスタンス、ディレクトリー、その他のサポートされるファイルを確立することが含まれます。ここでは、以下のトピックについて説明されます。

- 論理および物理データベースの特性の設計
- インスタンスの作成
- 環境変数およびプロファイル・レジストリー
- DB2 Administration Server
- ノード構成ファイルの作成
- データベース構成ファイルの作成
- 高速コミュニケーション・マネージャー (FCM) 通信

論理および物理データベースの特性の設計

データベースを作成する前に、論理的または物理的なデータベースの設計を決定しなければなりません。論理データベースと物理データベースの設計の詳細については、「*管理ガイド: プランニング*」を参照してください。

インスタンスの作成

インスタンスとは、データベースをカタログし、構成パラメーターを設定するための、論理データベース・マネージャー環境です。インスタンスは、必要に応じて複数作成できます。複数インスタンスを使用すると、次のことを行えます。

- 1 つのインスタンスを開発環境用に使用し、別のインスタンスを実稼働環境用に使用する。
- 特定の環境用にインスタンスを調整する。
- 機密情報へのアクセスを制限する。
- それぞれのインスタンスごとに SYSADM、SYSCTRL、および SYSMAINT 権限の割り当てを制御する。
- インスタンスごとにデータベース・マネージャーの構成を最適化する。
- インスタンスの失敗による影響を制限する。インスタンスが失敗した場合、1 つのインスタンスだけが影響を受けます。他のインスタンスは正常に機能し続けます。

注意点として、複数インスタンスには以下のように多少の欠点があります。

- インスタンスごとに、追加のシステム・リソース (仮想メモリーとディスク・スペース) が必要になる。
- 追加インスタンスを管理するためにさらに管理が必要になる。

インスタンス・ディレクトリーには、データベース・インスタンスに関連するすべての情報が保管されます。インスタンス・ディレクトリーの位置を作成後に変更することはできません。インスタンス・ディレクトリーの内容は、以下のとおりです。

- データベース・マネージャー構成ファイル

- システム・データベース・ディレクトリー
- ノード・ディレクトリー
- ノード構成ファイル (db2nodes.cfg)
- デバッグ情報 (例外ダンプ、レジスター・ダンプ、 DB2® Universal Database (DB2 UDB) プロセス用の呼び出しスタックなど) が入った他のファイル。

UNIX® オペレーティング・システムでは、インスタンス・ディレクトリーは INSTHOME/sqllib ディレクトリーにあります。ただし、INSTHOME は、インスタンス所有者のホーム・ディレクトリーです。

Windows® オペレーティング・システムでは、インスタンス・ディレクトリーは DB2 UDB インストール・ディレクトリー内の /sqllib サブディレクトリーに置かれます。

パーティション・データベース・システムにおけるインスタンス・ディレクトリーは、インスタンスに属するすべてのデータベース・パーティション・サーバー間で共用されます。したがって、インスタンス・ディレクトリーは、インスタンス内のすべてのマシンがアクセスできるネットワーク共用ドライブ上に作成しなければなりません。

インストール手順の一部として、「DB2」という DB2 UDB の初期インスタンスを作成します。 UNIX では、命名規則の指針にはずれない範囲で初期インスタンスの名前を付けることができます。インスタンス名は、ディレクトリー構造を設定するために使用します。

このインスタンスをすぐに使えるようにするために、インストール中に次の設定がなされます。

- 環境変数 DB2INSTANCE が ″DB2″ に設定される。
- DB2 レジストリー変数 DB2INSTDEF が ″DB2″ に設定される。

UNIX では、命名規則の指針にはずれない範囲でデフォルトの名前を付けることができます。

Windows ではインスタンス名がサービス名と同じなので、競合は発生しません。サービスを作成するための適切な許可が必要です。

これらの設定により、″DB2″ がデフォルト・インスタンスとして確立されます。デフォルトで使用されるインスタンスを変更することはできますが、最初に、追加インスタンスを作成する必要があります。

DB2 UDB を使用する前に、各ユーザーのデータベース環境を更新して、インスタンスにアクセスし、DB2 UDB プログラムを実行できるようにします。このことはすべてのユーザー (管理ユーザーも含む) に当てはまります。

UNIX オペレーティング・システムでは、データベース環境の設定に役立つサンプル・スクリプト・ファイルが提供されます。サンプル・ファイルは、Bourne または Korn シェルの場合は db2profile、 C シェルの場合は db2cshrc です。これらのスクリプトは、インスタンス所有者のホーム・ディレクトリー下の sqllib サブディレクトリーに配置されます。インスタンス所有者またはインスタンスの SYSADM

グループに属するユーザーはだれでも、インスタンスの全ユーザーのスクリプトをカスタマイズできます。また、スクリプトをユーザーごとにコピーして、カスタマイズすることもできます。

サンプル・スクリプトには、次の目的を持つステートメントが入っています。

- 既存の検索パスに以下のディレクトリーを追加して、ユーザーの「パス」を更新します。すなわち、インスタンス所有者のホーム・ディレクトリーの sqllib サブディレクトリー下にある bin、 adm、misc サブディレクトリー。
- DB2INSTANCE 環境変数をインスタンス名に設定します。

関連概念:

- 21 ページの『UNIX オペレーティング・システムでの複数インスタンス』
- 22 ページの『Windows オペレーティング・システムでの複数インスタンス』

関連タスク:

- 26 ページの『インスタンスの追加』
- 24 ページの『UNIX でのインスタンスの作成に関する詳細』
- 25 ページの『Windows でのインスタンスの作成に関する詳細』
- 27 ページの『現行インスタンスの設定』
- 28 ページの『インスタンスの自動開始』
- 28 ページの『複数のインスタンスの並行実行』
- 27 ページの『インスタンスのリスト』
- 23 ページの『追加のインスタンスの作成』

UNIX での DB2 UDB 環境の自動設定

デフォルトでは、インスタンス作成時にデータベース環境をセットアップするスクリプトは、現行セッションの期間中に限ってユーザー環境に影響を与えます。.profile ファイルを変更すれば、ユーザーが Bourne または Korn シェルを使ってログオンするときに、 db2profile スクリプトを自動的に実行できようになります。 C シェルのユーザーであれば、 .login ファイルを変更して db2shrc スクリプト・ファイルを実行できるようになります。

手順:

.profile または .login スクリプト・ファイルに、以下に示すいずれかのステートメントを追加します。

- 1 つのバージョンのスクリプトを共用するユーザーの場合は、次のように追加します。

```
. INSTHOME/sqllib/db2profile    (Bourne または Korn シェルの場合)
source INSTHOME/sqllib/db2cshrc (C シェルの場合)
```

  この場合の INSTHOME は、使用するインスタンスのホーム・ディレクトリーです。

- ホーム・ディレクトリーにカスタマイズしたバージョンのスクリプトがあるユーザーの場合は、次のように追加します。

```
. USERHOME/db2profile         (Bourne または Korn シェルの場合)
source USERHOME/db2cshrc      (C シェルの場合)
```

USERHOME は、ユーザーのホーム・ディレクトリーです。

関連タスク:

• 20 ページの『UNIX での DB2 UDB 環境の手動設定』

UNIX での DB2 UDB 環境の手動設定

手順:

使用したいインスタンスを選択するには、コマンド・プロンプトで次のいずれかのステートメントを入力します。ピリオド (.) とスペースは必須です。

- 1 つのバージョンのスクリプトを共用するユーザーの場合は、次のように追加します。

```
. INSTHOME/sqllib/db2profile    (Bourne または Korn シェルの場合)
source INSTHOME/sqllib/db2cshrc (C シェルの場合)
```

  この場合の INSTHOME は、使用するインスタンスのホーム・ディレクトリーです。

- ホーム・ディレクトリーにカスタマイズしたバージョンのスクリプトがあるユーザーの場合は、次のように追加します。

```
. USERHOME/db2profile         (Bourne または Korn シェルの場合)
source USERHOME/db2cshrc      (C シェルの場合)
```

  USERHOME は、ユーザーのホーム・ディレクトリーです。

同時に複数のインスタンスを処理したい場合は、使用する各インスタンスのスクリプトを別々のウィンドウで実行します。たとえば、test および prod という 2 つのインスタンスがあり、そのホーム・ディレクトリーが /u/test および /u/prod であるとします。

ウィンドウ 1 では、次のようにします。

- Bourne または Korn シェルでは、次のように入力します。

```
. /u/test/sqllib/db2profile
```

- C シェルでは、次のように入力します。

```
source /u/test/sqllib/db2cshrc
```

ウィンドウ 2 では、次のようにします。

- Bourne または Korn シェルでは、次のように入力します。

```
. /u/prod/sqllib/db2profile
```

- C シェルでは、次のように入力します。

```
source /u/prod/sqllib/db2cshrc
```

ウィンドウ 1 は test インスタンスを処理するため、ウィンドウ 2 は prod インスタンスを処理するために使用します。

注: which db2 コマンドを入力し、検索パスが正確に設定されていることを確かめます。このコマンドを実行すると、CLP 実行可能モジュールの絶対パスが戻されます。その位置が、インスタンスの sqllib ディレクトリーにあることを確かめてください。

関連タスク:

- 19 ページの『UNIX での DB2 UDB 環境の自動設定』

UNIX オペレーティング・システムでの複数インスタンス

1 つの UNIX® オペレーティング・システム上に複数のインスタンスを作成することが可能です。ただし、一度に 1 つの DB2® Universal Database (DB2 UDB) インスタンスでのみ作業できます。

注: 複数のインスタンス間での環境競合を防ぐために、各インスタンスごとにホーム・ファイル・システムを設定する必要があります。ホーム・ファイル・システムが共有されている場合、エラーが戻されます。

インスタンス所有者およびシステム管理 (SYSADM) グループであるグループは、個々のインスタンスと関連付けられます。インスタンス所有者および SYSADM グループは、インスタンスの作成プロセス中に割り当てられます。 1 つのユーザー ID またはユーザー名は 1 つのインスタンスでのみ使用することができます。そのユーザー ID またはユーザー名は、*インスタンス所有者* とも呼ばれます。

各インスタンス所有者はユニークなホーム・ディレクトリーを持つ必要があります。インスタンスの実行に必要なファイルはすべて、インスタンス所有者のユーザー ID またはユーザー名のホーム・ディレクトリーに作成されます。

インスタンス所有者のユーザー ID またはユーザー名をシステムから除去する必要が生じた場合、インスタンスに関連付けられたファイルと、そのインスタンスに保管されたデータへのアクセスを失うおそれがあります。このため、インスタンス所有者のユーザー ID またはユーザー名は、 DB2 UDB の実行専用で使用することをお勧めします。

インスタンス所有者の 1 次グループも重要です。この 1 次グループは自動的にインスタンスのシステム管理グループになり、インスタンスに対する SYSADM 権限を取得します。インスタンス所有者の 1 次グループのメンバーである他のユーザー ID またはユーザー名も、このレベルの権限を取得します。この理由から、インスタンス所有者のユーザー ID またはユーザー名は、インスタンスの管理用に確保した 1 次グループに割り当てたいと思うかもしれません。 (また、1 次グループは必ずインスタンス所有者のユーザー ID またはユーザー名に割り当てるようにします。割り当てなければ、システム・デフォルトの 1 次グループが使用されます。)

インスタンスのシステム管理グループにしたいグループがすでにある場合は、インスタンス所有者ユーザー ID またはユーザー名の作成時に、そのグループを 1 次グループとして割り当てるだけで済みます。インスタンスに対する管理権限を他のユーザーに付与するには、システム管理グループとして割り当てられたグループに該当するユーザーを追加します。

インスタンス間で SYSADM 権限を分離するには、インスタンス所有者ユーザー ID またはユーザー名ごとに異なる 1 次グループを使用します。ただし、複数インスタンスで共通の SYSADM 権限を持つことにした場合は、複数インスタンスに対して同じ 1 次グループを使用することができます。

関連タスク:

• 24 ページの『UNIX でのインスタンスの作成に関する詳細』

Windows オペレーティング・システムでの複数インスタンス

同じマシン上で、複数の DB2® Universal Database (DB2 UDB) インスタンスを実行することが可能です。それぞれの DB2 UDB インスタンスは独自のデータベースを保守し、独自のデータベース・マネージャー構成パラメーターを持っています。

各 DB2 UDB インスタンスは以下のもので構成されます。

- インスタンスを表す Windows® サービス。サービスの名前は、インスタンス名と同じです。 (「サービス」パネルでの) サービスの表示名は、インスタンス名の前にストリング ″DB2 - ″ が付きます。たとえば、インスタンス名が DB2 であれば、 ″DB2″ という名前の Windows サービスが存在し、その表示名は ″DB2 - DB2″ です。

  注: Windows サービスは、Windows 98、 Windows ME、およびクライアント・インスタンス用には作成されません。
- インスタンス・ディレクトリー。このディレクトリーには、データベース・マネージャー構成ファイル、システム・データベース・ディレクトリー、ノード・ディレクトリー、 DCS データベース・ディレクトリー、およびインスタンスに関連したすべての診断ログとダンプ・ファイルが含まれます。デフォルトでは、インスタンス・ディレクトリーは SQLLIB ディレクトリー内のサブディレクトリーとなり、インスタンス名と同じ名前になります。たとえば、インスタンス ″DB2″ のインスタンス・ディレクトリーは `C:¥SQLLIB¥DB2` です (`C:¥SQLLIB` は DB2 UDB のインストール場所)。レジストリー変数 DB2INSTPROF を使用して、インスタンス・ディレクトリーのデフォルト場所を変更することができます。レジストリー変数 DB2INSTPROF の値を他の場所に設定した場合、インスタンス・ディレクトリーは DB2INSTPROF で指定されたディレクトリーの下に作成されます。たとえば、`DB2INSTPROF=D:¥DB2PROFS` と設定した場合、インスタンス・ディレクトリーは `D:¥DB2PROFS¥DB2` となります。
- HKEY_LOCAL_MACHINE¥SOFTWARE¥IBM¥DB2¥PROFILES¥<instance_name> の下のレジストリー・キー。インスタンス・レベルのすべてのレジストリー変数がここに作成されます。

複数の DB2 UDB インスタンスを並行して実行することができます。特定の 1 つのインスタンスに関する作業を行うには、そのインスタンスに対してコマンドを発行する前に、DB2INSTANCE 環境変数をそのインスタンス名に設定する必要があります。

あるインスタンスが別のデータベース・インスタンスにアクセスするのを防ぐために、インスタンス用のデータベース・ファイルはインスタンス名と同じ名前のディレクトリーの下に作成されます。たとえば、インスタンス DB2 用のデータベースを `C:` ドライブに作成する場合、データベース・ファイルは `C:¥DB2` ディレクトリーの下に作成されます。同様に、インスタンス TEST 用のデータベースを `C:` ドライブに作成する場合、データベース・ファイルは `C:¥TEST` ディレクトリーの下に作成されます。

関連概念:

- 「*データ・リカバリーと高可用性 ガイドおよびリファレンス*」の『高可用性』

関連タスク:

- 25 ページの『Windows でのインスタンスの作成に関する詳細』

追加のインスタンスの作成

DB2 Universal Database™ (DB2 UDB) インストールの一部としてインスタンスが作成されますが、ビジネス・ニーズに応じて、さらに追加のインスタンスを作成する必要が生じる場合もあります。

前提条件:

Windows の管理グループに属しているユーザー、または UNIX プラットフォームの root 権限を持っているユーザーが、追加の DB2 UDB インスタンスを追加できます。インスタンスの追加先のマシンは、インスタンス所有マシン (ノード 0) になります。インスタンスは、DB2 Administration Server が常駐するマシン上に必ず追加してください。

手順:

コマンド行を使用してインスタンスを追加するには、以下のように入力します。

```
db2icrt <instance_name>
```

db2icrt コマンドを使用して別の DB2 UDB インスタンスを追加するときには、インスタンス所有者のログイン名を提供する必要があり、オプションで、インスタンスの認証タイプを指定します。認証タイプは、そのインスタンスの下で作成されたすべてのデータベースに適用されます。認証タイプとは、ユーザーの認証が行われる場所を示すものです。

インスタンス・ディレクトリーの位置は、 DB2INSTPROF 環境変数を使って DB2PATH から変更することができます。その場合は、インスタンス・ディレクトリーの書き込みアクセスが必要です。 DB2PATH 以外のパスにディレクトリーを作成したい場合、 **db2icrt** コマンドを入力する*前*に、 DB2INSTPROF を設定しなければなりません。

DB2 Universal Database Enterprise - Server Edition (ESE) の場合、追加する新規インスタンスがパーティション・データベース・システムであることを宣言する必要もあります。さらに、複数のパーティションがある ESE インスタンスに対して作業を行うとき、および高速コミュニケーション・マネージャー (FCM) に対して作業を行うときには、インスタンスの作成時に追加の TCP/IP ポートを定義することにより、パーティション相互間に複数の接続を確立できます。たとえば Windows オペレーティング・システムでは、 **db2icrt** コマンドに **-r <port range>** パラメーターを指定して使用します。ポートの範囲は以下のように示します。

```
-r:<base_port,end_port>
```

base_port は FCM が使用できる最初のポートで、 end_port は FCM が使用できるポート番号の範囲にある最後のポートです。

関連概念:

- 230 ページの『サーバーでの認証メソッド』
- 237 ページの『リモート・クライアントの認証に関する考慮事項』

関連資料:

- 「コマンド・リファレンス」の『db2icrt - インスタンスの作成コマンド』

UNIX でのインスタンスの作成に関する詳細

UNIX オペレーティング・システムで作業する場合、 **db2icrt** コマンドには、以下の任意指定パラメーターがあります。

- -h または -?

  このパラメーターは、コマンドのヘルプ・メニューを表示する場合に使用します。

- -d

  このパラメーターは、問題判別中に使用するデバッグ・モードを設定します。

- -a AuthType

  このパラメーターは、インスタンスの認証タイプを指定します。有効な認証タイプは、SERVER、SERVER_ENCRYPT、または CLIENT です。指定しない場合、DB2 Universal Database™ (DB2 UDB) サーバーがインストールされていれば、デフォルトは SERVER に設定されます。それ以外の場合は、CLIENT に設定されます。

  注:

  1. インスタンスの認証タイプは、インスタンスが所有するすべてのデータベースに適用されます。
  2. UNIX オペレーティング・システムでは、認証タイプ DCE の選択は無効です。

- -u FencedID

  このパラメーターは、 fenced ユーザー定義関数 (UDF) とストアード・プロシージャーを実行するユーザーです。 DB2 UDB クライアントまたは DB2 UDB Application Development Client をインストールする場合、これは必須ではありません。他の DB2 UDB 製品の場合、これは必須パラメーターです。

  注: FencedID は、″root″ または ″bin″ ではありません。

- -p PortName

  このパラメーターは、使用予定の TCP/IP サービス名またはポート番号を指定します。その際、この値はインスタンス内のすべてのデータベースについて、そのインスタンスのデータベース構成ファイルで設定されます。

- -s InstType

  異なるタイプのインスタンスを作成できます。有効なインスタンス・タイプは、`ese`、`wse`、 `client`、および `standalone` です。

次に例を示します。

- DB2 UDB サーバー用のインスタンスを追加するには、以下のコマンドを使用できます。

```
db2icrt -u db2fenc1 db2inst1
```

- DB2 Connect Enterprise Edition だけをインストールした場合は、インスタンス名を fenced ID としても使用できます。

  ```
  db2icrt -u db2inst1 db2inst1
  ```

- DB2 UDB クライアント用のインスタンスを追加するには、以下のコマンドを使用できます。

  ```
  db2icrt db2inst1 -s client -u fencedID
  ```

DB2 UDB クライアント・インスタンスは、ワークステーションから他のデータベース・サーバーへ接続したい場合に作成します。作成すると、そのワークステーションにローカル・データベースは必要なくなります。

関連資料:

- 「コマンド・リファレンス」の『db2icrt - インスタンスの作成コマンド』

Windows でのインスタンスの作成に関する詳細

Windows オペレーティング・システムで作業する場合、 **db2icrt** コマンドには、以下の任意指定パラメーターがあります。

- -s InstType

  異なるタイプのインスタンスを作成できます。有効なインスタンス・タイプは、`ese`、 `wse`、`client`、および `standalone` です。

- -p:InstProf_Path

  これは、別のインスタンス・プロファイル・パスを指定する任意指定パラメーターです。パスを指定しない場合、インスタンス・ディレクトリーは `SQLLIB` ディレクトリー内に作成され、インスタンス名に連結された共用名 `DB2` が指定されます。読み取りおよび書き込み許可は、ドメイン内の各ユーザーに自動的に与えられます。許可は、ディレクトリーへのアクセスを制限するために変更することができます。

  異なるインスタンス・プロファイル・パスを実際に指定する場合は、共用ドライブまたはディレクトリーを作成する必要があります。これにより、許可が変更されていなければ、ドメイン内の全員がインスタンス・ディレクトリーにアクセスできるようになります。

- -u:username,password

  パーティション・データベース環境を作成する場合、 DB2 Universal Database サービスのドメイン/ユーザー・アカウント名とパスワードを宣言しなければなりません。

- -r:base_port,end_port

  これは、高速コミュニケーション・マネージャー (FCM) の TCP/IP ポート範囲を指定する任意指定パラメーターです。 TCP/IP ポート範囲を指定する場合は、パーティション・データベース・システムのすべてのマシンで、そのポート範囲を使用できることを確認しなければなりません。

DB2 Universal Database (DB2 UDB) Enterprise Server Edition for Windows では、次の例を使用できます。

```
db2icrt inst1 -s ese
   -p:¥¥machineA¥db2mpp
   -u:<user account name>,<password> -r:9010,9015
```

注: サービス・アカウントを変更する場合 (つまり、製品のインストール時に最初に作成されたインスタンスであるデフォルト・サービスをもはや使用しない場合)、インスタンス作成の際に使用するドメイン/ユーザー・アカウント名に、以下のような拡張権限を付与する必要があります。

- オペレーティング・システムの一部として活動する。
- トークン・オブジェクトを作成する。
- 割り当て量を増やす。
- サービスとしてログオンする。
- 処理レベル・トークンを置換する。
- メモリー内のページをロックする。

インスタンスは、共用ドライブにアクセスし、ユーザー・アカウントを認証し、 DB2 UDB を Windows サービスとして実行するために、これらのユーザー権を必要とします。「メモリー内のページをロックする」権限は、 Address Windowing Extensions (AWE) サポートで必要とされます。

関連資料:

- 「コマンド・リファレンス」の『db2icrt - インスタンスの作成コマンド』

インスタンスの追加

手順:

追加のインスタンスを作成した後、そのインスタンスをコントロール・センターで扱うためには、そのインスタンスに関するレコードをコントロール・センターに追加する必要があります。

別のインスタンスを追加するには、以下のステップを実行します。

1. 管理権限を持っているか、またはローカル管理者グループに属するユーザー ID またはユーザー名でログオンします。
2. インスタンスを追加するには、次の方法のいずれかを使用してください。

   コントロール・センターを使用するには、以下のようにします。

1. オブジェクト・ツリーを順に展開し、使用するシステムの**「インスタンス (Instances)」**フォルダーを表示します。
2. インスタンスのフォルダーを右クリックして、ポップアップ・メニューから**「追加 (Add)」**を選択します。
3. 情報をすべて入力し、**「適用 (Apply)」**をクリックします。

関連概念:

- 17 ページの『インスタンスの作成』

関連タスク:

- 27 ページの『インスタンスのリスト』

インスタンスのリスト

手順:

コントロール・センターを使用して、システムで使用可能なインスタンスをすべてリストするには、次のようにします。

1. オブジェクト・ツリーを順に展開し、**「インスタンス (Instances)」**フォルダーを表示します。
2. インスタンスのフォルダーを右クリックして、ポップアップ・メニューから**「追加 (Add)」**を選択します。
3. 「インスタンスの追加 (Add Instance)」ウィンドウで、**「最新表示 (Refresh)」**をクリックします。
4. ドロップダウン矢印をクリックして、データベース・インスタンスのリストを表示します。
5. **「キャンセル (Cancel)」**をクリックして、ウィンドウを終了します。

コマンド行を使用して、システムで使用可能なインスタンスをすべてリストするには、次のようにします。

```
db2ilist
```

(サポートされている Windows プラットフォーム上で) 現行セッションに適用されるインスタンスを判別するには、次のコマンドを使用します。

```
set db2instance
```

関連資料:

- 「コマンド・リファレンス」の『db2ilist - インスタンスのリスト・コマンド』

現行インスタンスの設定

手順:

インスタンスのデータベース・マネージャーを開始または停止するためのコマンドを実行すると、 DB2 Universal Database™ (DB2 UDB) はそのコマンドを現行インスタンスに適用します。 DB2 UDB は以下のように現行インスタンスを判別します。

- 現行セッションに DB2INSTANCE 環境変数が設定されていれば、その値が現行インスタンスです。 DB2INSTANCE 環境変数を設定するには、次のように入力します。

  ```
  set db2instance=<new_instance_name>
  ```

- DB2INSTANCE 環境変数が現行セッション用に設定されていない場合、 DB2 UDB はシステム環境変数から DB2INSTANCE 環境変数用の設定値を使用します。 Windows NT では、システム環境変数はシステム環境内で設定されます。 Windows 9x では、`autoexec.bat` ファイルで設定されます。
- DB2INSTANCE 環境変数がまったく設定されていなければ、 DB2 UDB はレジストリー変数 DB2INSTDEF を使用します。

DB2INSTDEF レジストリー変数をレジストリーのグローバル・レベルで設定するには、次のように入力します。

```
db2set db2instdef=<new_instance_name> -g
```

現行セッションに適用されるインスタンスを判別するには、次のように入力します。

```
db2 get instance
```

関連タスク:

- 32 ページの『レジストリーおよび環境変数の宣言』

インスタンスの自動開始

手順:

Windows オペレーティング・システムでは、インストール時に作成される DB2 Universal Database™ (DB2 UDB) インスタンスはデフォルトで自動始動するよう設定されます。 **db2icrt** を使って作成されるインスタンスは、手動開始するよう設定されます。開始タイプを変更するには、「サービス」パネルに移動して、その DB2 UDB サービスのプロパティーを変更する必要があります。

UNIX オペレーティング・システムで、各システムの再始動後にインスタンスを自動始動できるようにするには、次のコマンドを入力します。

```
db2iauto -on <instance name>
```

<instance name> はインスタンスのログイン名です。

UNIX オペレーティング・システムで、各システムの再始動後にインスタンスを自動開始できないようにするには、次のコマンドを入力します。

```
db2iauto -off <instance name>
```

<instance name> はインスタンスのログイン名です。

関連概念:

- 17 ページの『インスタンスの作成』

関連資料:

- 「コマンド・リファレンス」の『db2iauto - Auto-start Instance コマンド』

複数のインスタンスの並行実行

手順:

コントロール・センターを使用して複数インスタンスを並行実行するには、以下のようにします。

1. オブジェクト・ツリーを順に展開し、**「データベース (Databases)」**フォルダーを表示します。
2. インスタンスを右クリックして、ポップアップ・メニューから**「開始 (Start)」**を選択します。
3. ステップ 2 を繰り返し実行し、並行実行させるインスタンスをすべて開始します。

(Windows のみ) コマンド行を使用して複数のインスタンスを並行実行するには、以下のようにします。

1. 次のように入力して、開始する他のインスタンスの名前に DB2INSTANCE 変数を設定します。

```
set db2instance=<another_instName>
```

2. **db2start** コマンドを入力してインスタンスを開始します。

関連概念:

- 6 ページの『データベース・マネージャーの複数インスタンス』

関連タスク:

- 24 ページの『UNIX でのインスタンスの作成に関する詳細』
- 25 ページの『Windows でのインスタンスの作成に関する詳細』
- 23 ページの『追加のインスタンスの作成』

ライセンス管理

DB2® Universal Database (DB2 UDB) 製品のライセンス管理は主に、製品のオンライン・インターフェースのコントロール・センター内でライセンス・センターを使って行います。ライセンス・センターでは、インストールした個々の製品に関して、ライセンス情報、統計、登録済みユーザー、および現行ユーザーを調べることができます。

コントロール・センターを使用できない場合、ライセンス管理ツール・コマンド **db2licm** を使って基本的なライセンス機能を実行できます。このコマンドを使用すれば、ローカル・システムにインストール済みのライセンスやポリシーを追加、削除、リスト、および変更することができます。

関連資料:

- 「コマンド・リファレンス」の『db2licm - ライセンス管理ツール・コマンド』

環境変数およびプロファイル・レジストリー

環境変数およびレジストリー変数は、データベース環境を制御します。

構成アシスタント (**db2ca**) を使って、構成パラメーターとレジストリー変数を構成することができます。

DB2® Universal Database (DB2 UDB) プロファイル・レジストリーの導入前は、(たとえば) Windows® ワークステーションの環境変数を変更するためには環境変数を変更してリブートする必要がありました。いくつかの例外はありますが、現在、環境は DB2 UDB プロファイル・レジストリーに保管されているレジストリー変数

によって制御されます。 UNIX® オペレーティング・システムで特定インスタンスに関するシステム管理 (SYSADM) 権限を持つユーザーは、そのインスタンスのレジストリー値を更新することができます。 Windows ユーザーの場合、レジストリー変数を更新するために SYSADM 権限は必要ありません。リブートせずにレジストリー変数を更新するには、 **db2set** コマンドを使用します。この情報は、即時にプロファイル・レジストリーの中に保管されます。DB2 UDB レジストリーは、変更を行った後に開始した DB2 UDB サーバー・インスタンスと DB2 UDB アプリケーションに更新情報を適用します。

レジストリーを更新する場合、現在実行中の DB2 UDB アプリケーションまたはユーザーに影響はありません。更新後に開始されたアプリケーションは新しい値を使用します。

注: DB2 UDB 環境変数 DB2INSTANCE および DB2NODE は、 DB2 UDB プロファイル・レジストリーに保管されない場合があります。一部のオペレーティング・システムでは、これらの環境変数を更新するために **set** コマンドを使用する必要があります。変更は、次回システムをリブートするまで有効です。 UNIX プラットフォームでは、 **set** コマンドの代わりに **export** コマンドを使用できます。

プロファイル・レジストリーを使用することによって、環境変数の中央制御が可能になります。さまざまなレベルのサポートが、さまざまなプロファイルを通して提供されるようになっています。環境変数のリモートでの管理も、DB2 Administration Server を使用すれば可能です。

以下の 4 つのプロファイル・レジストリーがあります。

- DB2 UDB インスタンス・レベル・プロファイル・レジストリー。 DB2 UDB 環境変数の大多数は、このレジストリーの中に置かれます。特定のインスタンスについての環境変数の設定値が、このレジストリーに保持されます。このレベルに定義された値は、グローバル・レベルの設定値をオーバーライドします。
- DB2 UDB グローバル・レベル・プロファイル・レジストリー。特定のインスタンスごとに 1 つの環境変数が設定されるのではない場合、このレジストリーが使用されます。このレジストリーは、マシン全体にわたる環境変数設定値を持ちます。 DB2 UDB ESE では、マシンごとに 1 つのグローバル・レベル・プロファイルが存在します。
- DB2 UDB インスタンス・ノード・レベル・プロファイル・レジストリー。このレジストリー・レベルには、複数パーティション環境のパーティション (ノード) に関する変数設定値が含まれます。このレベルで定義された値は、インスタンス・レベルおよびグローバル・レベルの設定値をオーバーライドします。
- DB2 UDB インスタンス・プロファイル・レジストリー。このレジストリーには、このシステムによって認識されるすべてのインスタンス名のリストが含まれます。 db2ilist を実行すれば、システムで使用できるすべてのインスタンスがリストされます。

DB2 UDB は、レジストリー値と環境変数をチェックし、それらを以下の順序で解決することによって、操作環境を構成します。

1. set コマンドを使用して設定された環境変数。 (あるいは、 UNIX プラットフォームでは export コマンド。)

2. インスタンス・ノード・レベル・プロファイルを使用して設定されたレジストリー値 (コマンド **db2set -i <instance name> <nodenum>** を使用)。
3. インスタンス・レベル・プロファイルを使用して設定されたレジストリー値 (コマンド **db2set -i** を使用)。
4. グローバル・レベル・プロファイルを使用して設定されたレジストリー値 (コマンド **db2set -g** を使用)。

インスタンス・レベル・プロファイル・レジストリー

パーティション・データベース環境で作業を行っている場合、UNIX と Windows では、いくつかの違いがあります。これらの違いについて、以下に例を示します。

「red」、「white」、および「blue」で識別される 3 つの物理ノードを持つ、パーティション・データベース環境があると想定します。 UNIX プラットフォームでは、インスタンス所有者がいずれかのノードから次を実行した場合、

```
db2set -i FOO=BAR
```

または

```
db2set FOO=BAR        ('-i' is implied)
```

FOO の値は現行インスタンスのすべてのノード (「red」、「white」、および「blue」) に対して可視になります。

UNIX プラットフォームでは、インスタンス・レベル・プロファイル・レジストリーは、 sqllib ディレクトリー内のテキスト・ファイルに保管されます。パーティション・データベース環境では、sqllib ディレクトリーはすべての物理ノードによって共用されているファイル・システムにあります。

Windows プラットフォームでは、ユーザーが「red」から同じコマンドを実行した場合、 FOO の値は現行インスタンスの「red」でのみ可視になります。 DB2 UDB はインスタンス・レベル・プロファイル・レジストリーを Windows レジストリーに保管します。物理ノード間に共用はありません。レジストリー変数をすべての物理マシンに設定するには、以下のように「rah」コマンドを使用します。

```
rah db2set -i FOO=BAR
```

rah はリモートで db2set コマンドを「red」、「white」、および「blue」上で実行します。

DB2REMOTEPREG を使用して、インスタンスを所有していないマシン上のレジストリー変数を構成することにより、インスタンスを所有しているマシン上のレジストリー変数を参照することができます。これにより、インスタンスを所有しているマシン上のレジストリー変数が、インスタンス内のすべてのマシン間で共用される環境が効果的に作成されます。

上に示された例を使用し、「red」が所有マシンと想定すると、次を行うことで、「red」上のレジストリー変数を共有するために、DB2REMOTEPREG が「white」および「blue」マシンに設定されます。

```
(on red) do nothing
(on white and blue) db2set DB2REMOTEPREG=¥¥red
```

DB2REMOTEPREG の設定値は、設定後に変更できません。

以下に REMOTEPREG がどのように動作するかを示します。

DB2 UDB が Windows でレジストリー変数を読み取る場合、最初に DB2REMOTEPREG 値を読み取ります。 DB2REMOTEPREG が設定されている場合、DB2REMOTEPREG 変数に指定されているマシン名のリモート・マシン上のレジストリーをオープンします。その後のレジストリー変数の読み取りおよび更新は、指定されたリモート・マシンにリダイレクトされます。

リモート・レジストリーへのアクセスには、ターゲット・マシンで Remote Registry Service が実行されている必要があります。また、ユーザーのログオン・アカウントおよびすべての DB2 UDB サービス・ログオン・アカウントに、リモート・レジストリーへの十分なアクセス権限が必要です。そのため、DB2REMOTEPREG を使用するには、Windows ドメイン環境で操作を行う必要があります。これによって、必要なレジストリー・アクセスが、ドメイン・アカウントに付与されます。

Microsoft® Cluster Server (MSCS) には考慮事項があります。 DB2REMOTEPREG を MSCS 環境では使用できません。すべてのマシンが同じ MSCS クラスターに属する MSCS 構成で稼働している場合、レジストリー変数はクラスター・レジストリー内に保持されます。そのため、それらはすでに同じ MSCS クラスター内のすべてのマシン間で共用されており、この場合は DB2REMOTEPREG を使用する必要はありません。

パーティションが複数の MSCS サーバーに及ぶ、マルチパーティション・フェイルオーバー環境で稼働している場合、インスタンスを所有しているマシンのレジストリー変数はこのクラスター・レジストリーにあるため、インスタンスを所有しているマシンを指すのに、DB2REMOTEPREG は使用できません。

関連概念:

- 「*管理ガイド: パフォーマンス*」の『DB2 レジストリー変数と環境変数』

関連タスク:

- 32 ページの『レジストリーおよび環境変数の宣言』

レジストリーおよび環境変数の宣言

手順:

db2set コマンドは、レジストリー変数 (および環境変数) のローカルでの宣言をサポートします。

コマンドに対するヘルプ情報を表示するには、以下を使用します。

```
db2set ?
```

サポートされるすべてのレジストリー変数の完全なセットをリストするには、以下を使用します。

```
db2set -lr
```

現在のインスタンスまたはデフォルトのインスタンスに定義されているすべてのレジストリー変数をリストするには、以下を使用します。

```
db2set
```

プロファイル・レジストリーに定義されているすべてのレジストリー変数をリストするには、以下を使用します。

```
db2set -all
```

現在のインスタンスまたはデフォルトのインスタンス内のレジストリー変数の値を表示するには、以下を使用します。

```
db2set registry_variable_name
```

全レベルのレジストリー変数の値を表示するには、以下を使用します。

```
db2set registry_variable_name -all
```

現在のインスタンスまたはデフォルトのインスタンス内のレジストリー変数の値を変更するには、以下を使用します。

```
db2set registry_variable_name=new_value
```

インスタンス内のすべてのデータベースについて、レジストリー変数のデフォルトを変更するには、以下を使用します。

```
db2set registry_variable_name=new_value
   -i instance_name
```

このインスタンス内の特定パーティションのレジストリー変数のデフォルトを変更するには、以下を使用します。

```
db2set registry_variable_name=new_value
   -i instance_name node_number
```

このシステム内のすべてのインスタンス内のレジストリー変数のデフォルトを変更するには、以下を使用します。

```
db2set registry_variable_name=new_value -g
```

Lightweight Directory Access Protocol (LDAP) を使用している場合、 LDAP で以下を使用してレジストリー変数を設定できます。

- LDAP でレジストリー変数をユーザー・レベルで設定するには、以下を使用します。

  ```
  db2set -ul
  ```

- LDAP でレジストリー変数をグローバル・レベルで設定するには、以下を使用します。

  ```
  db2set -gl user_name
  ```

LDAP 環境で実行している場合、ある DB2 Universal Database™ (DB2 UDB) レジストリー変数値については、LDAP 内で設定することができます。(つまり、その有効範囲が、ディレクトリー・パーティションまたは Windows NT ドメインに属するすべてのマシンとすべてのユーザーに渡る、グローバルなものとなります。) 現在のところ、LDAP グローバル・レベルで設定できる DB2 UDBレジストリー変数は、 DB2LDAP_KEEP_CONNECTION と DB2LDAP_SEARCH_SCOPE の 2 つのみです。

たとえば、LDAP のグローバル・レベルで検索の有効範囲の値を設定するには、以下を使用します。

```
db2set -gl db2ldap_search_scope = value
```

value には ″local″、″domain″、または ″global″ を指定できます。

注:

1. db2set コマンドによって DB2 UDB profile.env ファイルが同時に更新される場合 (つまり同じ瞬間、またはほとんど同じ瞬間に更新される場合)、profile.env ファイルのサイズは小さくなってゼロになります。 `db2set -all` の出力も矛盾する値を表示します。
2. マシン・グローバル・レベルで DB2 UDB レジストリー変数を設定するのに使われる `-g` オプションと、 LDAP グローバル・レベル固有の `-gl` の間には相違点があります。
3. ユーザー・レベルのレジストリー変数は、 LDAP 環境で Windows を実行する場合にのみサポートされます。
4. ユーザー・レベルの変数設定値には、ユーザー固有の変数設定値が含まれます。ユーザー・レベルでのすべての変更点は、LDAP ディレクトリーに書き込まれます。
5. パラメーター ″-i″、″-g″、″-gl″、″-ul″ は、 同じコマンド内では同時に使用できません。
6. 一部の変数は、デフォルトで常にグローバル・レベル・プロファイルに設定されます。そのようなパラメーターをインスタンスまたはノード・レベル・プロファイルで設定することはできません。たとえば、DB2SYSTEM および DB2INSTDEF があります。
7. UNIX では、インスタンスのレジストリー値を変更するためには、システム管理 (SYSADM) 権限を持っていなければなりません。グローバル・レベル・レジストリー中のパラメーターを変更できるのは、 root 権限を持つユーザーのみです。

1 つのインスタンスの 1 つのレジストリー変数をリセットして、グローバル・プロファイル・レジストリーの中のデフォルトに戻すには、以下を使用します。

```
db2set -r registry_variable_name
```

1 つのインスタンス内のノードの 1 つのレジストリー変数をリセットして、グローバル・プロファイル・レジストリーの中のデフォルトに戻すには、以下を使用します。

```
db2set -r registry_variable_name node_number
```

指定レベルの変数値を削除するには、同じコマンド構文を使って変数を設定できますが、変数値には何も指定しません。たとえば、ノード・レベルの変数設定値を削除するには、次のように入力します。

```
db2set registry_variable_name= -i instance_name
  node_number
```

変数値を削除してその使用を制限する場合、その値が上位のプロファイル・レベルで定義されていれば、次のように入力します。

```
db2set registry_variable_name= -null instance_name
```

このコマンドを使用すると、指定するパラメーターの設定値が削除され、上位レベルのプロファイルでこの変数の値 (この場合は DB2 UDB グローバル・レベル・プ

ロファイル) を変更できなくなります。ただし、指定する変数を下位レベルのプロファイル (この場合は DB2 UDB ノード・レベル・プロファイル) で引き続き設定することはできます。

関連概念:

- 「*管理ガイド: パフォーマンス*」の『DB2 レジストリー変数と環境変数』

関連タスク:

- 35 ページの『Windows 上の環境変数の設定』
- 37 ページの『UNIX システム上の環境変数の設定』
- 351 ページの『LDAP ディレクトリー・パーティションまたはドメインの検索』
- 353 ページの『LDAP 環境でのユーザー・レベルでの DB2 レジストリー変数の設定』

Windows 上の環境変数の設定

手順:

特定のレジストリー変数はすべて、 DB2 Universal Database™ (DB2 UDB) プロファイル・レジストリーで定義することをぜひお勧めします。 DB2 UDB 変数がレジストリー以外で設定されていれば、その変数をリモート管理することはできません。変数値を有効にするには、ワークステーションをリブートしなければなりません。

Windows オペレーティング・システムには 1 つのシステム環境変数 DB2INSTANCE があり、この値はプロファイル・レジストリーの外部でしか設定できません。ただし、DB2INSTANCE の設定は必須ではありません。 DB2 UDB プロファイル・レジストリー変数 DB2INSTDEF をグローバル・レベル・プロファイルで設定すれば、 DB2INSTANCE が定義されていない場合に使用するインスタンス名を指定することができます。

Windows 上の DB2 UDB Enterprise Server Edition サーバーには、プロファイル・レジストリーの外部でしか設定できない 2 つのシステム環境変数 DB2INSTANCE および DB2NODE があります。 DB2INSTANCE の設定は必須ではありません。 DB2 UDB プロファイル・レジストリー変数 DB2INSTDEF をグローバル・レベル・プロファイルで設定すれば、 DB2INSTANCE が定義されていない場合に使用するインスタンス名を指定することができます。

DB2NODE 環境変数は、マシン内のターゲット論理ノードへの要求を経路指定するために使用します。この環境変数は、DB2 UDB プロファイル・レジストリーの中ではなく、アプリケーションまたはコマンドが発行されるセッションで設定しなければなりません。この変数を指定しない場合、ターゲット論理ノードはデフォルトとして、マシン上でゼロ (0) に定義された論理ノードに設定されます。

環境変数の設定値を判別するためには、**echo** コマンドを使用します。たとえば、DB2PATH 環境変数の値を検査するには、以下のように入力します。

```
echo %db2path%
```

システム環境変数を設定するには、以下のことを行います。

Windows 9x の場合: autoexec.bat ファイルを編集し、変更を有効にするためにシステムをリブートします。

Windows の場合: DB2 UDB 環境変数 DB2INSTANCE および DB2NODE を次のように設定できます (以下では DB2INSTANCE について説明します)。

- (Windows NT および Windows 2000 の場合) **「スタート」**、**「設定」**、**「コントロール パネル」**を選択します。 (Windows XP および Windows Server 2003 の場合) **「スタート」** —> **「コントロール パネル」**を選択します。
- (Windows NT および Windows 2000 の場合) **「システム」**アイコンをダブルクリックします。 (Windows XP および Windows Server 2003 の場合) Windows テーマ、および現在選択されているビュー・タイプに応じて、**「システム」**アイコンを選択する前に、**「パフォーマンス」**および**「メンテナンス」**を選択しなければならない場合もあります。
- (Windows NT の場合) 「コントロール パネル」の「システム」の「システム環境変数」セクションで、以下のようにします。(Windows 2000、Windows XP、および Windows Server 2003 の場合) 「システムのプロパティ」ウィンドウの**「詳細設定」**タブを選択し、「環境変数」ボタンをクリックして以下のようにします。
  1. DB2INSTANCE 変数が存在しない場合は、以下のようにします。
     a. (Windows NT の場合) 任意のシステム環境変数を選択します。 (Windows 2000、Windows XP、および Windows Server 2003 の場合) **「新規」**ボタンをクリックします。
     b. (Windows NT の場合) 「*変数*」フィールド内の名前を DB2INSTANCE に変更します。 (Windows 2000、Windows XP、および Windows Server 2003 の場合) 「*変数名*」フィールドに DB2INSTANCE と入力します。
     c. (Windows NT の場合) 「*値*」フィールドをインスタンス名 (たとえば、db2inst) に変更します。 (Windows 2000、Windows XP、および Windows Server 2003 の場合) 「*変数の値*」フィールドにインスタンス名 (たとえば db2inst) を入力します。
  2. DB2INSTANCE 変数がすでに存在している場合は、以下のようにして新しい値を追加します。
     a. DB2INSTANCE 環境変数を選択します。
     b. 「*値*」フィールドをインスタンス名 (たとえば、db2inst) に変更します。
  3. (Windows NT の場合)「設定」を選択します。 (Windows 2000、Windows XP、および Windows Server 2003 の場合) 「OK」を選択します。
  4. 「OK」を選択します。
  5. これらの変更が有効になるよう、システムをリブートします。

注: 環境変数 DB2INSTANCE もセッション (プロセス) レベルで設定できます。たとえば、TEST という 2 番目の DB2 UDB インスタンスを開始する場合、コマンド・ウィンドウで以下のコマンドを発行します。

```
set DB2INSTANCE=TEST
db2start
```

C シェルのコマンド・ウィンドウから、以下のコマンドを発行します。

```
setenv DB2INSTANCE TEST
```

プロファイル・レジストリーは、以下のように配置されます。

- Windows オペレーティング・システム・レジストリーの中の DB2 UDB インスタンス・レベル・プロファイル・レジストリーは、以下のパスを使用して配置されます。

  ¥HKEY_LOCAL_MACHINE¥SOFTWARE¥IBM¥DB2¥PROFILES¥*instance_name*

  注: *instance_name* は、DB2 UDB インスタンスの名前です。
- Windows レジストリーの中の DB2 UDB グローバル・レベル・プロファイル・レジストリーは、以下のパスを使用して配置されます。

  ¥HKEY_LOCAL_MACHINE¥SOFTWARE¥IBM¥DB2¥GLOBAL_PROFILE
- Windows レジストリーの中の DB2 UDB インスタンス・ノード・レベル・プロファイル・レジストリーは、以下のパスを使用して配置されます。

  ...¥SOFTWARE¥IBM¥DB2¥PROFILES¥*instance_name*¥NODES¥*node_number*

  注: *instance_name* と *node_number* は、作業しているデータベース・パーティションに固有のものです。
- DB2 UDB インスタンス・プロファイル・レジストリーは必要ありません。システム内の各 DB2 UDB インスタンスごとに、1 つのキーが以下のパスに作成されます。

  ¥HKEY_LOCAL_MACHINE¥SOFTWARE¥IBM¥DB2¥PROFILES¥*instance_name*

  インスタンスのリストを表示するには、PROFILES キーの下にあるキーを数えます。

関連概念:

- 45 ページの『DB2 Administration Server』

関連タスク:

- 37 ページの『UNIX システム上の環境変数の設定』

UNIX システム上の環境変数の設定

手順:

特定のレジストリー変数はすべて、DB2 UDB プロファイル・レジストリーで定義することをぜひお勧めします。 DB2 UDB 変数がレジストリー以外で設定されていれば、その変数をリモート管理することはできません。

UNIX オペレーティング・システムでは、システム環境変数 DB2INSTANCE を設定しなければなりません。

db2profile (Korn シェルの場合) および db2cshrc (Bourne シェルまたは C シェルの場合) というスクリプトが、データベース環境のセットアップを援助するために例として提供されています。これらのファイルは insthome/sqllib の中にあります (ただし、 insthome はインスタンス所有者のホーム・ディレクトリーです)。

これらのスクリプトには、以下のためのステートメントが入っています。

- 以下のディレクトリーにユーザーのパスを更新します。
  - insthome/sqllib/bin

  - insthome/sqllib/adm
  - insthome/sqllib/misc
- 実行のために、 DB2INSTANCE をデフォルトのローカル instance_name に設定します。

注: PATH および DB2INSTANCE を除いて、サポートされる変数は DB2 UDB プロファイル・レジストリーで設定しなければなりません。 DB2 UDB で設定されない変数を設定するには、スクリプト・ファイル userprofile および usercshrc で定義する必要があります。

インスタンス所有者または SYSADM ユーザーは、インスタンスのすべてのユーザーのためにこれらのスクリプトをカスタマイズすることができます。あるいは、ユーザーがスクリプトをコピーしてカスタマイズした後、スクリプトを直接呼び出すか、または自分の .profile または .login ファイルに追加することができます。

現行セッションについての環境変数を変更するには、以下のようなコマンドを出します。

- Korn シェルの場合、

```
DB2INSTANCE=inst1
   export DB2INSTANCE
```

- Bourne シェルの場合、

```
export DB2INSTANCE=<inst1>
```

- C シェルの場合、

```
setenv DB2INSTANCE <inst1>
```

DB2 UDB プロファイル・レジストリーが正しく管理されるようにするためには、UNIX オペレーティング・システム上で、以下のファイル所有権規則に従わなければなりません。

- DB2 UDB インスタンス・レベル・プロファイル・レジストリー・ファイルは、以下のもとに配置されます。

  ```
  INSTHOME/sqllib/profile.env
  ```

  このファイルのアクセス許可と所有権は、以下のようにする必要があります。

  ```
  -rw-rw-r-- <db2inst1> <db2iadm1> profile.env
  ```

  ここで、<db2inst1> はインスタンス所有者、 <db2iadm1> はインスタンス所有者のグループです。

  INSTHOME は、インスタンス所有者のホーム・パスです。
- DB2 UDB グローバル・レベル・プロファイル・レジストリーは、以下のもとに配置されます。
  - AIX、Solaris オペレーティング環境、および Linux オペレーティング・システムの場合、 /var/db2/<version_id>/default.env (<version_id> は現行バージョン)。
  - HP-UX オペレーティング・システムの場合は、/var/opt/db2/<version_id>/default.env (<version_id> は現行バージョン)。

  このファイルのアクセス許可と所有権は、以下のようにする必要があります。

```
-rw-rw-r-- <Instance_Owner> <Instance_Owner_Group> default.env
```

グローバル・レジストリー変数を修正する場合、ユーザーは root としてログオンしなければなりません。

- DB2 UDB インスタンス・ノード・レベル・プロファイル・レジストリーは、以下のもとに配置されます。

  ```
  INSTHOME/sqllib/nodes/<node_number>.env
  ```

  ディレクトリーとこのファイルのアクセス許可と所有権は、以下のようにする必要があります。

  ```
  drwxrwsr-w <Instance_Owner> <Instance_Owner_Group> nodes

  -rw-rw-r-- <Instance_Owner> <Instance_Owner_Group> <node_number>.env
  ```

  INSTHOME は、インスタンス所有者のホーム・パスです。

- DB2 UDB インスタンス・プロファイル・レジストリーは、以下のもとに配置されます。
  - AIX、Solaris、および Linux オペレーティング・システムの場合、/var/db2/<version_id>/profiles.reg (<version_id> は現行バージョン)。
  - HP-UX オペレーティング・システムの場合は、/var/opt/db2/<version_id>/profiles.reg (<version_id> は現行バージョン)。

  このファイルのアクセス許可と所有権は、以下のようにする必要があります。

  ```
  -rw-r--r-- root system profiles.reg
  ```

関連概念:

- 45 ページの『DB2 Administration Server』

関連タスク:

- 35 ページの『Windows 上の環境変数の設定』

ノード構成ファイルの作成

手順:

データベースをパーティション・データベース環境で操作する場合、 db2nodes.cfg というノード構成ファイルを作成する必要があります。このファイルは、並列機能を持ったデータベース・マネージャーを複数パーティションにわたって開始する前に、そのインスタンスに対するホーム・ディレクトリーの sqllib サブディレクトリーの中に配置されていなければなりません。このファイルには、1 つのインスタンスの中のすべてのデータベース・パーティションの構成情報が含まれ、そのインスタンスのすべてのデータベース・パーティションによって共用されます。

Windows での考慮事項

DB2 Universal Database™ (DB2 UDB) Enterprise Server Edition を Windows で使用している場合、インスタンス作成時にノード構成ファイルが作成されます。ノード構成ファイルは手動で作成したり変更したりしないでください。 **db2ncrt** コマンドを使用して、データベース・パーティション・サーバーをインスタンスに追加することができます。 **db2ndrop** コマンドを使

用して、データベース・パーティション・サーバーをインスタンスからドロップすることができます。 **db2nchg** コマンドを使用すれば、データベース・パーティション・サーバーの構成を変更することができます。たとえば、1 つのマシンから別のマシンへのデータベース・パーティション・サーバーの移動、 TCP/IP ホスト名の変更、または別の論理ポートやネットワーク名の選択を行うことができます。

注: インスタンスが削除された場合にデータの消失を避けるため、 sqllib サブディレクトリーには、 DB2 UDB によって作成されたもの以外のファイルまたはディレクトリーを作成しないでください。ただし、以下の 2 つの例外があります。システムがストアード・プロシージャーをサポートしている場合は、ストアード・プロシージャー・アプリケーションを sqllib サブディレクトリーの下の function サブディレクトリーに入れます。もう 1 つの例外は、ユーザー定義関数 (UDF) が作成される場合です。 UDF の実行可能コードは、同じディレクトリーに入れることが許されます。

ファイルには、1 つのインスタンスに属する各データベース・パーティションごとに 1 行が含まれます。それぞれの行は、以下の形式になっています。

```
dbpartitionnum hostname [logical-port [netname]]
```

トークンはブランクで区切られます。変数は、以下のとおりです。

dbpartitionnum
ノードを固有に定義するデータベース・パーティション番号 (0 から 999 まで)。データベース・パーティション番号は、昇順でなければなりません。間の番号が抜けていてもかまいません。

いったんデータベース・パーティション番号が割り当てられると、それを変更することはできません。 (変更すると、データをパーティション化する方法を指定するパーティション・マップの中の情報が信用できないものになります。)

ノードをドロップした場合、そのデータベース・パーティション番号は、追加する任意の新しいノード用に再使用することができます。

データベース・パーティション番号は、データベース・ディレクトリー内にノード名を生成するために使用されます。ノード名は、以下の形式になります。

```
NODEnnnn
```

nnnn はデータベース・パーティション番号で、左側はゼロで埋められます。このデータベース・パーティション番号は、 **CREATE DATABASE** コマンドおよび **DROP DATABASE** コマンドでも使用されます。

hostname
パーティション間通信のための IP アドレスのホスト名。ホスト名には、完全修飾名を使用します。 /etc/hosts ファイルも、完全修飾名を使用する必要があります。 db2nodes.cfg ファイルおよび /etc/hosts ファイルで完全修飾名を使用しない場合、エラー・メッセージ SQL30082N RC=3 を受け取る場合があります。

(netname が指定された場合は、例外です。この場合、netname がほとんどの通信で使用され、 hostname は **db2start**、**db2stop**、および **db2_all** でのみ使用されます。)

logical-port
このパラメーターの指定は任意であり、ノードの論理ポート番号を指定します。この番号は、データベース・マネージャー・インスタンス名と一緒に使用され、 etc/services ファイルの中の TCP/IP サービス名項目を識別します。

IP アドレスと論理ポートの組み合わせは、既知のアドレスとして使用され、ノード間の通信接続をサポートするために、すべてのアプリケーション内でユニークなものでなければなりません。

各 hostname について、1 つの logical-port は、 0 かまたはブランク (0 がデフォルト) でなければなりません。この logical-port に関連付けられるノードは、クライアントが接続するホスト上のデフォルトのノードです。これは、db2profile スクリプト内の DB2NODE 環境変数か、または sqlesetc() API を使用してオーバーライドすることができます。

同じホスト上に複数のノードがある場合 (つまり、 1 つのホストに対して複数の dbpartitionnum がある場合) は、 logical-port 番号を論理ノードに、0 から間の番号を抜かずに割り当ててください。順番は重要ではありません。

たとえば、以下のセットアップは有効です。

```
0   cpaiss43.mach1.xxx.com   1
1   cpaiss43.mach1.xxx.com   0
2   cpaiss43.mach1.xxx.com   2
3   cpaiss44.mach1.xxx.com   0
```

netname
このパラメーターの指定は任意であり、それぞれが独自のホスト名を持つ、複数のアクティブな TCP/IP インターフェースを持ったホストをサポートするために使用されます。

以下の例は、RS/6000 SP システムのための可能なノード構成ファイルを示したものであり、 SP2EN1 には複数の TCP/IP インターフェースと 2 つの論理パーティションがあり、 DB2 UDB のインターフェースとして SP2SW1 を使用しています。この例は、パーティション番号が (0 ではなく) 1 から始まり、 dbpartitionnum の順番は間の番号が抜けていることも示しています。

表 1. データベース・パーティション番号の例の表

dbpartitionnum	hostname	logical-port	netname
1	SP2EN1.mach1.xxx.com	0	SP2SW1
2	SP2EN1.mach1.xxx.com	1	SP2SW1
4	SP2EN2.mach1.xxx.com	0	
5	SP2EN3.mach1.xxx.com		

好みのエディターを使用して、db2nodes.cfg ファイルを更新することができます。(例外: Windows ではエディターは使用しないでください。) ただし、データ・パーティションではデータベース・パーティション番号を変更してはならないため、フ

ファイル内の情報の保全性を注意して保護する必要があります。ノード構成ファイルは、**db2start** を出したときにロックされ、 **db2stop** がデータベース・マネージャーを停止させた後でアンロックされます。ファイルがロックされている場合、**db2start** コマンドで、必要に応じて、ファイルを更新することができます。たとえば、RESTART オプションまたは ADDNODE オプションを指定して **db2start** を出すことができます。

注: db2stop コマンドが失敗し、ノード構成ファイルがアンロックされない場合、アンロックするために **db2stop FORCE** を発行してください。

関連概念:

- 「*管理ガイド: パフォーマンス*」の『ストアード・プロシージャーのガイドライン』

関連資料:

- 「コマンド・リファレンス」の『db2start - DB2 の開始コマンド』
- 「コマンド・リファレンス」の『db2stop - DB2 の停止コマンド』
- 「コマンド・リファレンス」の『CREATE DATABASE コマンド』
- 「コマンド・リファレンス」の『DROP DATABASE コマンド』
- 「コマンド・リファレンス」の『db2nchg - データベース・パーティション・サーバー構成の変更コマンド』
- 「コマンド・リファレンス」の『db2ncrt - インスタンスへのデータベース・パーティション・サーバーの追加コマンド』
- 「コマンド・リファレンス」の『db2ndrop - インスタンスからのデータベース・パーティション・サーバーのドロップ・コマンド』

データベース構成ファイルの作成

手順:

データベースごとにデータベース*構成ファイル* が作成されます。このファイルの作成は管理者のために行われます。このファイルには、データベースの使用に影響を与える、次のような様々な*構成*パラメーター の値が入れられます。

- データベースの作成時に指定または使用されるパラメーター (データベース・コード・ページ、照合順序、DB2 UDB リリース・レベルなど)
- データベースの現行の状態を示すパラメーター (バックアップ・ペンディング・フラグ、データベース一貫性フラグ、ロールフォワード操作ペンディング・フラグなど)
- データベースの操作時に使用されるシステム・リソースの量を定義するパラメーター (バッファー・プール・サイズ、データベース・ロギング、ソート・メモリー・サイズなど)

構成ファイル内のパラメーターは手動で変更しないでください。サポートされているインターフェースだけを使用する必要があります。

パフォーマンス上のヒント: 構成パラメーターの多くはデフォルト値が提供されていますが、データベースの最適なパフォーマンスを達成するためには構成パラメーターの更新が必要な場合があります。

複数パーティションの場合: 複数のパーティションにわたってパーティション化されたデータベースを持っている場合、構成ファイルは、すべてのデータベース・パーティションで同じものである必要があります。 SQL コンパイラーは、ローカル・ノードの構成ファイルの情報に基づいて分散 SQL ステートメントをコンパイルし、 SQL ステートメントのニーズを満足させるためのアクセス・プランを作成するので、これらの構成ファイルには整合性が必要です。データベース・パーティションごとに異なる構成ファイルを維持していると、どのデータベース・パーティションでステートメントが準備されたかによって、異なるアクセス・プランが作成される可能性があります。 **db2_all** を使用して、すべてのデータベース・パーティションで構成ファイルの同期を保ってください。

関連概念:

- 379 ページの『パーティション・データベース環境でのコマンドの実行』

関連タスク:

- 「*管理ガイド: パフォーマンス*」の『構成パラメーターによる DB2 の構成』

高速コミュニケーション・マネージャー (FCM) 通信

パーティション・データベース環境では、データベース・パーティション間のほとんどの通信は、高速コミュニケーション・マネージャー (FCM) によって処理されます。データベース・パーティションで FCM を使用可能にし、他のデータベース・パーティションとの通信ができるようにするには、下記に示すように、パーティションの `etc` ディレクトリーの `services` ファイル内にサービス項目を作成する必要があります。FCM は、指定されたポートを使用して通信を行います。同じホスト上に複数のパーティションを定義している場合、以下に示すように、ある範囲のポートを定義しなければなりません。

Windows® での考慮事項

DB2® Universal Database (DB2 UDB) Enterprise Server Edition を Windows 環境で使用している場合、 TCP/IP のポート範囲は次のものによって自動的にサービス・ファイルに追加されます。

- インストール・プログラムがインスタンスを作成したり新しいノードを追加したりするときに、インストール・プログラムによって
- **db2icrt** ユーティリティーが新しいインスタンスを作成するときに、db2icrt ユーティリティーによって
- **db2ncrt** ユーティリティーがマシンに最初のノードを追加したときに、db2ncrt ユーティリティーによって

サービス項目の構文は、以下のとおりです。

```
DB2_instance port/tcp #comment
```

DB2_*instance*

instance の値は、データベース・マネージャー・インスタンスの名前です。名前の中のすべての文字は小文字でなければなりません。 `db2puser` というインスタンス名であるとすれば、 `DB2_db2puser` というように指定します。

port/**tcp**

データベース・パーティションのために予約したい TCP/IP ポート。

#*comment*

この項目と関連付けたい任意の注釈。注釈の前には、ポンド記号 (#) を付けなければなりません。

etc ディレクトリーの services ファイルが共用されている場合、ファイル内に割り当てられるポートの数は、そのインスタンス内の複数のデータベース・パーティションの最大数と等しいかそれより大きくなるようにしなければなりません。ポートを割り当てる場合には、バックアップとして使用できるプロセッサーもその数の中に入れるようにしなければなりません。

etc ディレクトリーの services ファイルが共用されていない場合、同じ考慮事項が適用されます。ただし追加の考慮事項として、DB2 UDB インスタンス用に定義される項目は、 etc ディレクトリーの services ファイルで同じでなければなりません (パーティション・データベースに適用されない他の項目は、同じである必要はありません)。

1 つのインスタンス内で同じホスト上に複数のデータベース・パーティションがある場合、使用する FCM のために複数のポートを定義しなければなりません。そのためには、etc ディレクトリーの services ファイルの中に 2 行を組み込んで、割り当てるポートの範囲を示します。最初の行は最初のポートを指定し、2 番目の行は複数のポート・ブロックの終わりを示します。以下の例では、sales というインスタンスに 5 つのポートが割り当てられます。これは、そのインスタンスには、 5 つを超えるデータベース・パーティションを持つプロセッサーはないことを意味します。たとえば、次のようにします。

```
DB2_sales         9000/tcp
DB2_sales_END     9004/tcp
```

注: END は、大文字でのみ指定しなければなりません。また、両方の下線 (_) 文字も含めるようにしなければなりません。

関連概念:

- 「*管理ガイド: プランニング*」の『パーティションおよびプロセッサー環境』

第 2 章 DB2 Administration Server (DAS) の作成と使用

DB2 Administration Server (DAS) は、 DB2 サーバー作業を支援するために使われます。

DB2 Administration Server

DB2® Administration Server (DAS) は、DB2 Universal Database™ DB2 UDB サーバーでの作業を専門的に支援するためのコントロール・ポイントです。構成アシスタント、コントロール・センター、または デベロップメント・センターなどのツールを使用するには、 DAS を実行していなければなりません。 DAS は、以下の管理タスクを処理するときにコントロール・センターおよび構成アシスタントを援助します。

- DB2 UDB サーバーをリモートに管理できる。
- DB2 UDB とオペレーティング・システム・コマンド・スクリプトの両方の実行をスケジュールする機能も含めた、ジョブ管理用の機能を提供する。これらのコマンド・スクリプトはユーザーが定義します。
- タスク・センターを使用して、ジョブ・スケジュールの定義、完了したジョブの結果表示、および DAS にとってリモートまたはローカルに置かれているジョブに対する他の管理タスクの実行を行います。
- DB2 UDB ディスカバリー・ユーティリティーと共に、 DB2 UDB インスタンス、データベースおよび他の DB2 Administration Server の構成についての情報を検出するための手段を提供する。この情報は、DB2 UDB データベースへのクライアント接続の構成を単純化して自動化するために、構成アシスタントとコントロール・センターが使用します。

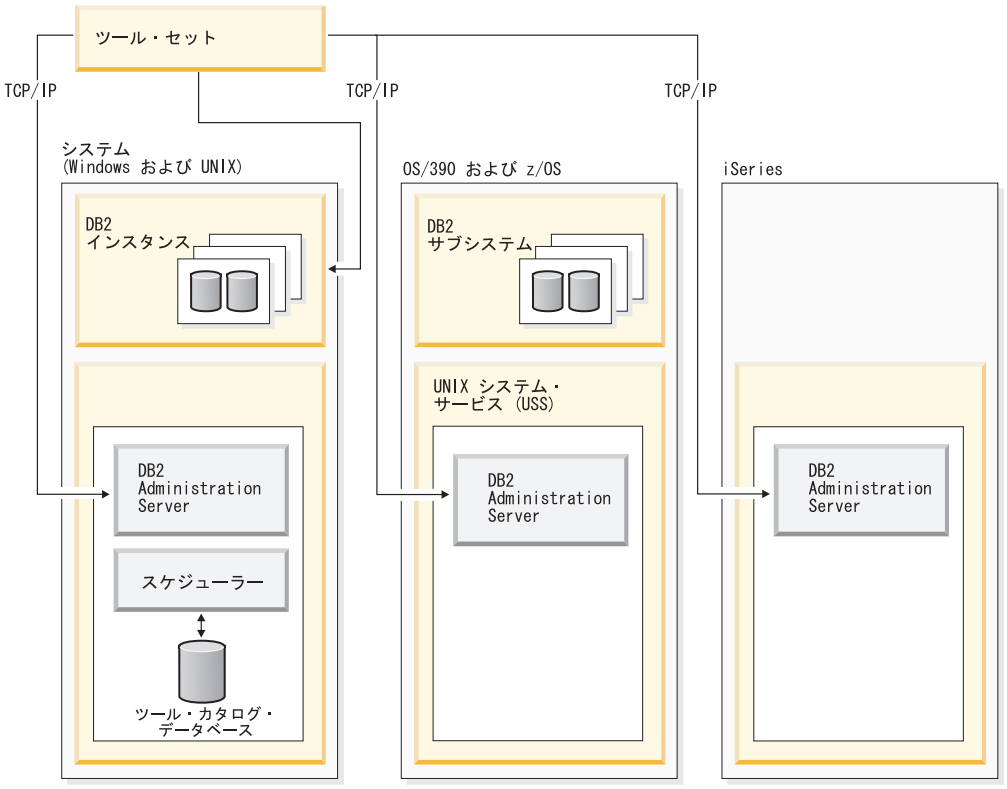


図 1. DAS が使用される場所

1 つのマシン上には 1 つの DAS しか持つことができません。 DAS はインストール中に構成され、オペレーティング・システムのブート時に開始します。

DAS の用途は、コントロール・センター、構成アシスタント、その他のツールからのクライアント要求を扱うために、サーバー・システムおよびホスト・システムでリモート・タスクを実行することです。

DAS は、zSeries® (OS/390 および z/OS™ のみ) プラットフォームだけではなく、サポートされるすべての Windows® および UNIX® で使用できます。 zSeries 上で DAS を使用すれば、コントロール・センター、デベロップメント・センター、およびレプリケーション・センターの管理作業がサポートされます。

zSeries (OS/390 および z/OS のみ) 用の DB2 Administration Server は、 DB2 UDB 管理のクライアント機能の一部としてパッケージされ、提供されます。 DAS を必要とする製品 (コントロール・センター、レプリケーション・センター、デベロップメント・センターなど) では、DAS 機能をインストールする必要があります。ご使用のオペレーティング・システムでの DAS の利用について、詳しくは IBM® 担当員にお問い合わせください。

Windows および UNIX 用の DAS には、タスク・センターを使って定義されるタスク (たとえば、 DB2 UDB およびオペレーティング・システムのコマンド・スクリプト) を実行するためのスケジューラーが含まれます。実行されるコマンドなどに関するタスク情報、すなわちタスクに関連したスケジュール、通知、および完了アクション、およびタスクの実行結果が、ツール・カタログと呼ばれる DB2 UDB データベース内の表およびビューのセットに保管されます。ツール・カタログは、

セットアップの一部として作成されます。さらに、コントロール・センターで、あるいは CLP で **CREATE TOOLS CATALOG** コマンドを使って、これを作成して活動化することもできます。

スケジューラーは zSeries (OS/390 および z/OS のみ) では提供されませんが、コントロール・センターの「JCL のビルド」および「JCL の作成」機能を使用して JCL を生成し、パーティション・データ・セットに保管して、システム・スケジューラーで実行することができます。

関連概念:

- 58 ページの『Windows での DAS のセキュリティーに関する考慮事項』
- 63 ページの『Enterprise Server Edition (ESE) システムでの DAS 構成』
- 64 ページの『Administration Server、インスタンス、およびデータベースのディスカバリー』
- 69 ページの『DB2 Administration Server での First Failure Data Capture (FFDC)』

関連タスク:

- 47 ページの『DB2 Administration Server の作成』
- 48 ページの『DAS の開始と停止』
- 50 ページの『DAS のリスト』
- 50 ページの『DAS の構成』
- 59 ページの『UNIX での DAS の更新』
- 60 ページの『DAS の除去』
- 61 ページの『Enterprise Server Edition (ESE) システムでの DAS の設定』
- 66 ページの『ディスカバリーからサーバー・インスタンスおよびデータベースを隠す』
- 67 ページの『ディスカバリー・パラメーターの設定』
- 68 ページの『構成アシスタントおよびコントロール・センターを使用するための DAS の設定』
- 68 ページの『ディスカバリー用の DAS 構成の更新』
- 51 ページの『ツール・カタログ・データベース、DAS スケジューラーのセットアップ、および構成』
- 57 ページの『通知、連絡先リストのセットアップ、および構成』
- 57 ページの『DAS Java 仮想マシンのセットアップ』

DB2 Administration Server の作成

DB2 Administration Server は、コントロール・センターや構成アシスタントなど、DB2 Universal Database™ (DB2 UDB) ツール用のサポート・サービスを提供します。

前提条件:

DAS を作成するには、UNIX プラットフォームの root 権限、またはサービスを作成するための適切な許可が必要です。

Windows で特定のユーザーが識別される場合には、ローカル管理者権限を持つユーザーを作成してください。 **db2admin create** を入力します。特定のユーザー・アカウントを使用する場合には、 **db2admin create** を発行するときに、 ″/USER:″ と ″/PASSWORD:″ を使用する必要があります。

手順:

一般にセットアップ・プログラムは、 DB2 UDB インストール中にインスタンス所有マシン上に DAS を作成します。しかし、セットアップ・プログラムが DAS の作成に失敗した場合は、 DAS を手動で作成することができます。

インストール処理の間に DAS に関連して生じることの概要については、以下のことを考慮してください。

- Windows プラットフォームでは、以下のようにします。

  サービスを作成するための適切な許可を持つアカウントを使用して、 DAS を作成するマシンにログオンします。

  DAS を作成するときに、ユーザー・アカウント名とユーザー・パスワードを指定することができます (指定は任意です)。有効であれば、ユーザー・アカウント名とパスワードは、DAS の所有者を識別します。 DAS 用に作成したユーザー ID またはアカウント名は、ユーザー・アカウントとして使用しないでください。アカウント名のパスワードを ″Password Never Expires″ (無期限のパスワード) に設定します。 DAS を作成した後で、**db2admin setid** コマンドを使用してユーザー・アカウント名とユーザー・パスワードを提供することによって、その所有権を確立または修正することができます。
- UNIX プラットフォームでは、以下のようにします。
  1. root 権限を持っていることを確認します。
  2. コマンド・プロンプトで、以下のコマンドを DB2 UDB インストール・パスの `instance` サブディレクトリーから発行します。

     ```
     dascrt -u <DASUser>
     ```

     `<DASUser>` は、 DB2 UDB ユーザーおよびグループの作成時に作成された DAS ユーザーの名前です。

     – AIX では、次のようにします。

       ```
       /usr/opt/db2_08_01/instance/
       dascrt -u <DASUser>
       ```

     – HP-UX、Solaris オペレーティング環境、または Linux では、次のようにします。

       ```
       /opt/IBM/db2/V8.1/instance/
       dascrt -u <DASUser>
       ```

関連資料:

- 「コマンド・リファレンス」の『db2admin - DB2 Administration Server コマンド』

DAS の開始と停止

手順:

Windows 上で DAS を手動で開始または停止するには、最初に、管理者、サーバー演算子、またはパワー・ユーザーのいずれかのグループに属するアカウントまたはユーザー ID を使ってマシンにログオンしなければなりません。 UNIX で DAS を手動で開始または停止するには、アカウントまたはユーザー ID が *dasadm_group* に属している必要があります。 *dasadm_group* は、DAS 構成パラメーターで指定されます。

Windows で DAS を開始または停止するには、 **db2admin start** コマンドまたは **db2admin stop** コマンドを使用します。

DB2 Universal Database™ (DB2 UDB) (任意の UNIX オペレーティング・システム用) で作業する場合は、以下のことを行う必要があります。

- DAS を開始するには、以下のようにします。
  1. DAS 所有者としてログインします。
  2. 次のいずれか 1 つを使用してスクリプトのセットアップを実行します。

     ```
     . DASHOME/das/dasprofile    (Bourne または Korn シェルの場合)
     source DASHOME/das/dascshrc (C シェルの場合)
     ```

     ここで、DASHOME は DB2 Administration Server のホーム・ディレクトリーです。
  3. 次のように **db2admin** コマンドを使って DAS を開始します。

     ```
     db2admin start
     ```

  注: DAS は各システムのリブート後に自動的に開始されます。デフォルトでの DAS の始動時の動作は、 **dasauto** コマンドを使って変更できます。
- DAS を停止するには、以下のようにします。
  1. *dasadm_group* に属するアカウントまたはユーザー ID を使用してログインします。
  2. 次のように **db2admin** コマンドを使って DAS を停止します。

     ```
     db2admin stop
     ```

注: UNIX のもとでは、上記の両方の場合に、これらのコマンドを使用する人は、DAS 所有者の許可 ID を使用してログオンされていなければなりません。 **db2admin start** コマンドまたは **db2admin stop** コマンドを発行するには、ユーザーが *dasadm_group* に属していなければなりません。

関連資料:

- 「コマンド・リファレンス」の『db2admin - DB2 Administration Server コマンド』
- 「*管理ガイド: パフォーマンス*」の『dasadm_group - 「DAS 管理者権限グループ名」構成パラメーター』

DAS のリスト

手順:

マシン上の DAS の名前を取得するには、次のように入力します。

```
db2admin
```

このコマンドは、DAS の開始と停止、新しいユーザーとパスワードの作成、DAS のドロップ、 DAS に関連したユーザー・アカウントの確立や修正にも使用します。

関連資料:

- 「*コマンド・リファレンス*」の『db2admin - DB2 Administration Server コマンド』

DAS の構成

手順:

DAS に関連した DB2 Administration Server 構成パラメーターの現行値を表示するには、以下を入力します。

```
db2 get admin cfg
```

このコマンドを実行すると、製品のインストール中にデフォルトとして与えられた現行値、または構成パラメーターを以前更新したときに与えられた現行値が表示されます。

コマンド行プロセッサー (CLP) および UPDATE ADMIN CONFIG コマンドを使用して DAS 構成を更新するには、 DAS と同じインストール済みレベルにあるインスタンスから CLP を使用する必要があります。 DAS 構成ファイルの個々の項目を更新するには、以下を入力します。

```
db2 update admin cfg using ...
```

構成パラメーターを推奨されるデフォルトにリセットするには、以下を入力します。

```
db2 reset admin cfg
```

場合によっては、DAS 構成ファイルの変更内容は、それらがメモリーにロードされた後に有効になります (つまり、 **db2admin stop** の後に **db2admin start** が実行されたとき、あるいは Windows プラットフォームの場合はサービスを停止して開始したとき。) その他の場合、構成パラメーターはオンラインで構成可能です。(つまり、パラメーターを有効にするために DAS を再始動する必要はありません。)

関連タスク:

- 「*管理ガイド: パフォーマンス*」の『構成パラメーターによる DB2 の構成』

関連資料:

- 「*コマンド・リファレンス*」の『UPDATE ADMIN CONFIGURATION コマンド』

ツール・カタログ・データベース、DAS スケジューラーのセットアップ、および構成

ツール・カタログ・データベースには、タスク・センターおよびコントロール・センターによって作成されたタスク情報が含まれます。これらのタスクは、DB2 Administration Server のスケジューラーによって実行されます。スケジューラーとツール・カタログ・データベースは、常に共に機能します。どちらも他方がなくては機能しません。スケジューラーは DB2 Administration Server の特定の部分であり、ツール・カタログ・データベースを読み取るエージェントとして機能して、タスクをそれぞれの決まった時間に実行します。

前提条件:

DB2 Administration Server がインストール済みでなければなりません。

手順:

ツール・カタログ・データベースおよび DAS スケジューラーをセットアップして構成します。

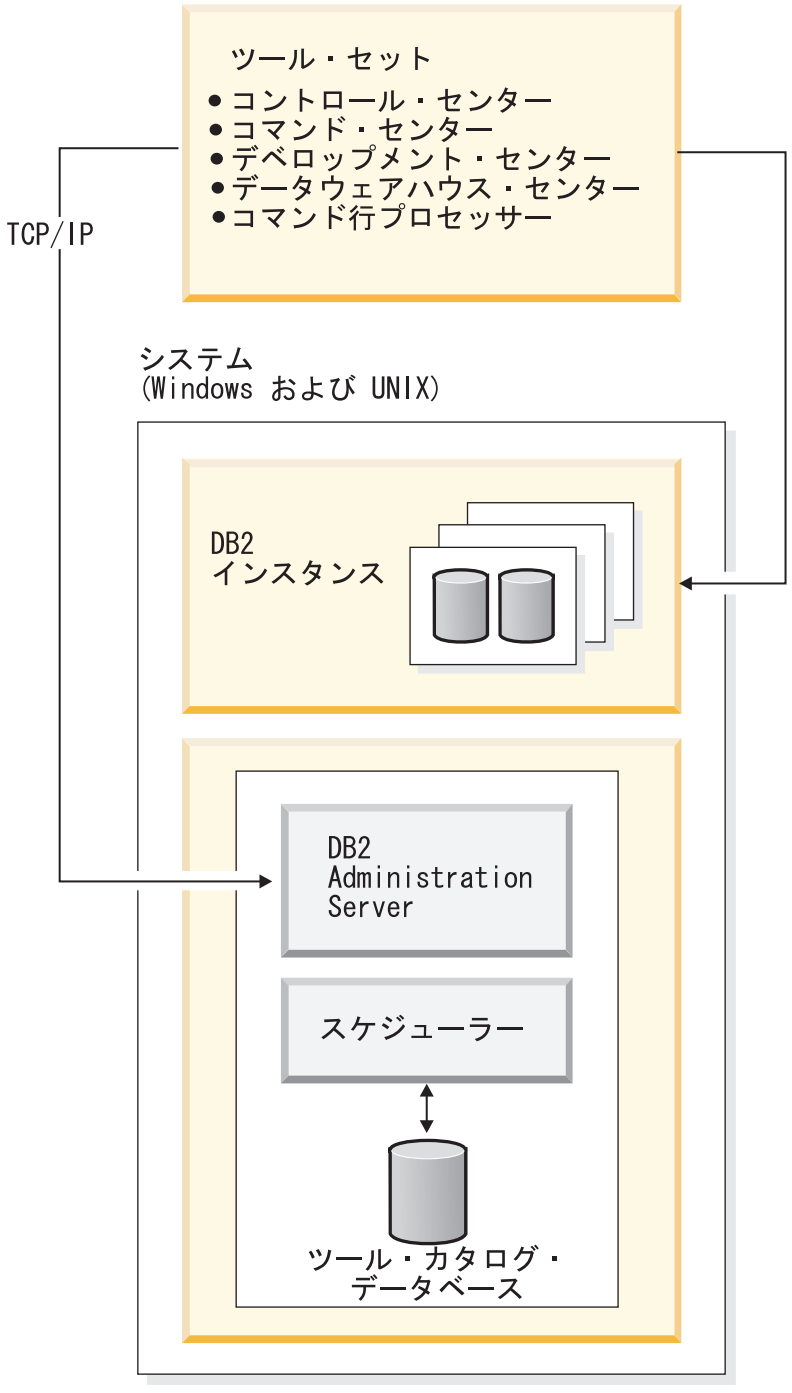


図 2. DAS と DB2 UDB の他の部分との関連

DB2 Administration Server の構成プロセスはスケジューラーに対し、ツール・カタログ・データベースの場所、およびスケジューラーを使用可能にするかどうかを知らせます。デフォルトでは、ツール・カタログ・データベースの作成時に、対応する DAS 構成が更新されます。つまり、スケジューラーは構成されて新しいツール・カタログを使用できるようになります。 DAS を再始動する必要はありません。

ツール・カタログ・データベースは、スケジューラー・システムに対してローカルまたはリモートの場所にあるサーバー上に作成できます。ツール・カタログをリモート・サーバー上に作成する場合、それをスケジューラーのツール・カタログ・データベースのインスタンス (TOOLSCAT_INST) でカタログする必要があります。さらに、コマンド **db2admin setschedid** を使用してスケジューラーのユーザー ID を設定し、スケジューラーがリモート・カタログに接続してそれを認証できるようにする必要があります。 **db2admin** コマンドの完全な構文については、「コマンド・リファレンス」を参照してください。

DAS スケジューラーは、ツール・カタログ情報にアクセスするために Java 仮想マシン (JVM) を必要とします。 JVM 情報を指定するには、 DAS に関する DB2 Administration Server 構成パラメーター `jdk_path` を使用します。

32 ビットおよび 64 ビット・インスタンスの両方をサポートするいずれかのプラットフォーム (AIX、Sun、および HP-UX) 上の 64 ビット・インスタンスに対してツール・カタログを作成する場合には、 `jdk_64_path` 構成パラメーターが必要です。

コントロール・センターおよびタスク・センターは、ツール・カタログ・データベースにクライアントから直接アクセスします。したがって、コントロール・センターで使用する前に、ツール・カタログ・データベースをクライアントでカタログしておく必要があります。コントロール・センターは、ツール・カタログ・データベースに関する情報を自動的に検索し、ローカル・ノード・ディレクトリーおよびデータベース・ディレクトリー内に必要なディレクトリー項目を作成します。この自動カタログ用にサポートされる通信プロトコルは、TCP/IP のみです。

DAS パラメーターの 1 つは *exec_exp_task* です。このパラメーターは、すでにスケジュール済みの、まだ実行されていないタスクをスケジューラーが実行するかどうかを指定します。スケジューラーの始動時には、有効期限切れタスクを検出するだけです。

たとえば、毎週土曜日に実行するようにスケジュールされたジョブがあり、スケジューラーが金曜日にオフにされ、月曜日に再始動された場合、土曜日にスケジュールされていたジョブは、過去にスケジュールされていたジョブになります。 *exec_exp_task* が「Yes」に設定されている場合、土曜日のジョブは、スケジューラーが再始動した時点で実行されます。

スケジューラーが必要とするその他の DAS 構成パラメーターは、ツール・カタログ・データベースの識別と、通知に使われる Simple Mail Transfer Protocol (SMTP) サーバーの識別で構成されます。

以下の例は、これらのパラメーターの使用法を示しています。

- Windows サーバー・セットアップの例
  1. ツール・カタログ・データベースは任意の名前にすることができます。この例では、ツール・カタログ・データベースの名前は ″CCMD″ で、サーバー・マシン Host1 (tcp/ip hostname Host1) 上の DB2 Universal Database™ (DB2 UDB) インスタンスの下に作成されます。特定のデータベース内のツール・カタログを固有に識別するために、スキーマ名を使用します。この例では、スキーマ名を「CCADMIN」と想定します。

2. 「DB2」というインスタンスは、ポート番号 50000 を使用して、以下のように TCP/IP 通信用にセットアップされます。

```
db2set -i DB2 DB2COMM=TCPIP
db2 update dbm cfg using svcename db2cDB2
db2stop
db2start
```

3. サービス名 db2cDB2 は、 %SystemRoot%¥system32¥drivers¥etc¥services で定義されます。つまり、services の中に以下の 1 行があります。

```
db2cDB2     50000/tcp     #connection port for the DB2 instance DB2
```

4. ツール・カタログは、CREATE TOOLS CATALOG コマンドを使用して作成されます。これにより、指定したデータベース内のカタログ名と対応するスキーマ名を持つ、ツール・カタログ表およびビューが作成されます。 DB2 Administration Server 構成パラメーターは自動的に更新され、スケジューラーが使用可能になり、開始されます。

5. e-mail 通知に使われる SMTP サーバーがマシン Host2 (tcp/ip hostname Host2) 上にあるとします。以下を使用して、この情報が DB2 Administration Server に対して指定されます。

```
db2 update admin cfg using smtp_server Host2
```

これは、インストール中に行われる場合があります。そうでない場合は、上記のような DB2 UDB バージョン 8 CLP コマンドを使用して、手動で DAS に対して指定する必要があります。

6. Windows 上の IBM Software Development Kit (SDK) for Java は、%DB2PATH%¥java¥jdk の下にインストールされます。これは、DAS に対してすでに指定済みでなければなりません。必要であれば、以下のようにして、このパラメーターを設定します。

```
db2 update admin cfg using jdk_path c:¥SQLLIB¥java¥jdk
```

ここでは、DB2 UDB が C:¥SQLLIB の下にインストールされていることを想定しています。

注: DAS を db2admin create で作成する場合、 /USER および /PASSWORD オプションを必ず使用してください。 USER アカウントは、スケジューラー・プロセスによって使用されます。これがないと、スケジューラーは正常に開始されません。 USER アカウントには、ツール・カタログ・インスタンスで SYSADM 権限が必要です。

DAS が db2admin create で作成され、/USER および /PASSWORD オプションが同時に指定されていない場合は、後で USER 情報を更新できます。この更新は、以下のコマンドを実行することで、DAS 上で行えます。

```
db2admin stop
db2admin setid <user account ID> <password>
db2admin start
```

- Windows クライアント・セットアップの例

1. コントロール・センターがクライアント・マシン C1 (tcp/ip ホスト名 C1) で実行されているとします。

2. DAS は、ローカル・ノード・ディレクトリー内の Administration Server ノードとしてカタログされます。そうするには、構成アシスタントまたはコントロール・センターを使用するか、以下のコマンドを使用します。

   ```
   db2 catalog admin tcpip node Host1 remote Host1 system Host1 ostype NT
   ```

3. タスク・センターが開始済みで、システム Host1 が選択されている場合、タスク・センターはローカル・ディレクトリー内にツール・カタログ・データベースを見つけようとします。 (タスク・センターの代わりにコントロール・センターを使用することもできます。) 見つからない場合、以下を使用して、ノードとデータベースをカタログしようとします。

   ```
   db2 catalog tcpip node <unique-node name>
     remote Host1 server 50000
     remote_instance DB2 system Host1 ostype NT
   db2 catalog db CCMD as <unique-db alias> at node <unique-node name>
   ```

   自動カタログが失敗した場合、構成アシスタントまたはコントロール・センターを使ってデータベースをカタログすることができます。その後、データベースはタスク・センターに認識され、使用できるようになります。

- AIX サーバー・セットアップの例

  1. ツール・カタログ・データベースは任意の名前にすることができます。この例では、ツール・カタログ・データベースの名前は ″CCMD″ で、サーバー・マシン Host1 (tcp/ip hostname Host1) 上の db2inst1 インスタンスの下に作成されます。特定のデータベース内のツール・カタログを固有に識別するために、スキーマ名を使用します。この例では、スキーマ名を「CCADMIN」と想定します。
  2. インスタンス db2inst1 は、ポート番号 50000 を使用して、以下のように TCP/IP 通信用にセットアップされます。

     ```
     db2set -i DB2 DB2COMM=TCPIP
     db2 update dbm cfg using svcename xdb2inst
     db2stop
     db2start
     ```

  3. xdb2inst サービス名は、 /etc/services に定義されています。つまり、services の中に以下の 1 行があります。

     ```
     xdb2inst1     50000/tcp     #connection port for the DB2 instance db2inst1
     ```

  4. ツール・カタログは、CREATE TOOLS CATALOG コマンドを使用して作成されます。これにより、指定したデータベース内のカタログ名と対応するスキーマ名を持つ、ツール・カタログ表およびビューが作成されます。 DB2 Administration Server 構成パラメーターは自動的に更新され、スケジューラーが使用可能になり、開始されます。
  5. e-mail 通知に使われる SMTP サーバーがマシン Host2 (tcp/ip hostname Host2) 上にあるとします。以下を使用して、この情報が DB2 Administration Server に対して指定されます。

     ```
     db2 update admin cfg using smtp_server Host2
     ```

     これは、インストール中に行われる場合があります。そうでない場合は、上記のような DB2 UDB バージョン 8 CLP コマンドを使用して、手動で DAS に対して指定する必要があります。

6. AIX 上の IBM Software Developer's Kit for Java (SDK) バージョン 1.3.1 は、 /sqllib/java/jdk の下にインストールされます。これは、DAS に対してすでに指定済みでなければなりません。必要であれば、以下のようにして、このパラメーターを設定します。

   ```
   db2 update admin cfg using jdk_path /sqllib/java/jdk
   ```

- AIX クライアント・セットアップの例
  1. コントロール・センターがクライアント・マシン C1 (tcp/ip ホスト名 C1) で実行されているとします。
  2. DAS は、ローカル・ノード・ディレクトリー内の Administration Server ノードとしてカタログされます。そのためには、構成アシスタントまたはコントロール・センターを使用するか、以下のコマンドを使用します。

     ```
     db2 catalog admin tcpip node Host1 remote Host1 system Host1
       ostype AIX
     ```

  3. タスク・センターが開始済みで、システム Host1 が選択されている場合、タスク・センターはローカル・ディレクトリー内にツール・カタログ・データベースを見つけようとします。 (タスク・センターの代わりにコントロール・センターを使用することもできます。) 見つからない場合、以下を使用して、ノードとデータベースをカタログしようとします。

     ```
     db2 catalog tcpip node <unique-node name>
       remote Host1 server 50000
       remote_instance DB2 system Host1 ostype AIX
     db2 catalog db CCMD as <unique-db alias> at node <unique-node name>
     ```

     自動カタログが失敗した場合、構成アシスタントまたはコントロール・センターを使ってデータベースをカタログすることができます。その後、データベースはタスク・センターに認識され、使用できるようになります。

関連資料:

- 「*管理ガイド: パフォーマンス*」の『svcename - 「TCP/IP サービス名」構成パラメーター』
- 「*管理ガイド: パフォーマンス*」の『sched_enable - 「スケジューラー・モード」構成パラメーター』
- 「*管理ガイド: パフォーマンス*」の『toolscat_inst - 「ツール・カタログ・データベース・インスタンス」構成パラメーター』
- 「*管理ガイド: パフォーマンス*」の『toolscat_db - 「ツール・カタログ・データベース」構成パラメーター』
- 「*管理ガイド: パフォーマンス*」の『toolscat_schema - 「ツール・カタログ・データベース・スキーマ」構成パラメーター』
- 「*管理ガイド: パフォーマンス*」の『smtp_server - 「SMTP サーバー」構成パラメーター』
- 「*管理ガイド: パフォーマンス*」の『jdk_path - 「Java Development Kit インストール・パス DAS」構成パラメーター』
- 「*管理ガイド: パフォーマンス*」の『exec_exp_task - 「有効期限切れタスクの実行」構成パラメーター』

通知、連絡先リストのセットアップ、および構成

DB2 Administration Server (DAS) からの E メールおよびページャー通知は、ローカルまたはリモートで行うことができます。通知が確実に正しいホスト名に送られるようにするには、連絡先リストが必要です。

手順:

スケジューラーまたはヘルス・モニターによる通知を使用可能にする DAS 構成パラメーターには、 2 つあります。

DAS 構成パラメーター *smtp_server* を使って、スケジューラーによって使用される Simple Mail Transfer Protocol (SMTP) サーバーを識別します。スケジューラーはこのサーバーを使用し、タスク・センターまたはヘルス・モニターで定義された e-mail またはページャー通知の送信方法に従って、タスク実行完了アクションの一部として e-mail やページャーでアラート通知を送ります。

DAS 構成パラメーター *contact_host* は、スケジューラーやヘルス・モニターが通知のために使用する連絡先情報の保管場所を指定します。この場所は、DB2 Administration Server の TCP/IP ホスト名として定義されます。 *contact_host* をリモート DAS に置くと、複数の DB2 Administration Server 間での連絡先リストの共用がサポートされます。パーティション・データベース環境では、すべてのパーティションが共通の連絡先リストを使用するように、これを設定する必要があります。連絡先リストは、DAS ディレクトリーの下にフラット・ファイルとして保管されます。 *contact_host* が指定されない場合、 DAS は連絡先情報がローカルに指定されているものと見なします。

関連資料:

- 「*管理ガイド: パフォーマンス*」の『smtp_server - 「SMTP サーバー」構成パラメーター』
- 「*管理ガイド: パフォーマンス*」の『contact_host - 「連絡先リストのロケーション」構成パラメーター』

DAS Java 仮想マシンのセットアップ

手順:

jdk_path 構成パラメーターは、 DB2 Administration Server 関数を実行するときに使用される IBM Software Developer's Kit (SDK) for Java のインストール・ディレクトリーを指定します。 Java インタープリターで使用される環境変数は、このパラメーターの値から計算されます。

スケジューラーは、ツール・カタログ・データベースを使用するために Java 仮想マシン (JVM) を必要とします。スケジューラーが正常に開始されるためには、あらかじめこれをセットアップしておく必要があります。

UNIX プラットフォームの場合、このパラメーターにデフォルト値はありません。IBM Software Developer's Kit (SDK) for Java のインストール時にこのパラメーターの値を指定する必要があります。

Windows 上の IBM Software Developer's Kit (SDK) for Java は、%DB2PATH%¥java¥jdk (つまり、 Windows プラットフォームでのこのパラメーターのデフォルト値) にインストールされます。これは、すでに DAS に対して指定されていなければなりません。 *jdk_path* の値を検査するには、以下のようにします。

```
db2 get admin cfg
```

このコマンドによって DB2 Administration Server 構成ファイルの値がすべて表示され、構成パラメーター *jdk_path* がその中に含まれます。必要であれば、以下のようにして、このパラメーターを設定できます。

```
db2 update admin cfg using jdk_path 'C:¥Program Files¥IBM¥SQLLIB'
```

ここでは、DB2 Universal Database™ (DB2 UDB) が C:¥Program Files¥IBM¥SQLLIB にインストールされていることを想定しています。

AIX 上の IBM Software Developer's Kit (SDK) for Java は、 /usr/java130 の下にインストールされます。必要であれば、以下のようにして、このパラメーターを設定できます。

```
db2 update admin cfg using jdk_path /usr/java130
```

注: 32 ビットおよび 64 ビット・インスタンスの両方をサポートするいずれかのプラットフォーム (AIX、Sun、または HP-UX) 上の 64 ビット・インスタンスに対してツール・カタログを作成または使用する場合には、 jdk_path パラメーターの代わりに jdk_64_path 構成パラメーターを使用してください。この構成パラメーターは、 64 ビット・バージョンの IBM Software Develop's Kit (SDK) for Java がインストールされているディレクトリーを指定します。

関連資料:

- 「コマンド・リファレンス」の『GET ADMIN CONFIGURATION コマンド』
- 「コマンド・リファレンス」の『UPDATE ADMIN CONFIGURATION コマンド』
- 「*管理ガイド: パフォーマンス*」の『jdk_path - 「Java Development Kit インストール・パス DAS」構成パラメーター』

Windows での DAS のセキュリティーに関する考慮事項

DAS サービスが Windows® で実行するためのユーザー ID を変更する必要があるかもしれません。

DAS を作成したら、次のように **db2admin** コマンドを使って、ログオン・アカウントを設定または変更できます。

```
db2admin setid <username> <password>
```

ここで、<username> と <password> は、ローカル管理者権限を持つアカウントのユーザー名とパスワードです。このコマンドの実行前に、ローカル管理者権限を持つアカウントまたはユーザー ID を使ってマシンにログオンしなければなりません。

注: パスワードが大文字小文字を区別することを忘れないでください。大文字と小文字を混合することができますが、パスワードの大文字小文字の区別が重要になります。

注: Windows では、ログオン・アカウント用に設定されない必須アクセス権があるので、**「コントロール パネル」**の**「サービス」**ユーティリティーを使って DAS 用のログオン・アカウントを変更しないでください。 DB2® Administration Server (DAS) 用のログオン・アカウントを設定または変更する場合は必ず、 **db2admin** コマンドを使用します。

関連資料:

- 「コマンド・リファレンス」の『db2admin - DB2 Administration Server コマンド』

UNIX での DAS の更新

手順:

UNIX オペレーティング・システムでは、 DB2 をプログラム一時修正 (PTF) またはコード・パッチによって更新する場合、各 DB2 Administration Server (DAS) とインスタンスを更新する必要があります。 DAS を更新するには、インストールした DB2 バージョンとリリースに固有のサブディレクトリー下の `instance` サブディレクトリーにある **dasupdt** コマンドを使用します。

最初に、スーパーユーザー権限 (通常 ″root″) を使ってマシンにログオンしなければなりません。

コマンドは、次のように使用します。

```
dasupdt
```

このコマンドに関連して、以下のオプショナル・パラメーターもあります。

- -h または -?

  このコマンドのヘルプ・メニューを表示します。

- -d

  デバッグ・モードを設定し、問題分析に使用します。

- -D

  1 つのパスのより高いコード・レベルから、別のパスにインストールされたより低いコード・レベルに DAS を移します。

注: Windows では、DAS の更新はインストール・プロセスの一部です。ユーザー処置は必要ありません。

例:

DAS がバージョン 8 インストール・パス内のバージョン 8.1.2 コードを実行しています。バージョン 8 インストール・パスにフィックスパック 3 がインストール済みの場合、バージョン 8 インストール・パスから以下のコマンドを呼び出すことにより、 DAS をフィックスパック 3 に更新します。

```
dasupdt
```

DAS が代替インストール・パス内のバージョン 8.1.2 コードを実行しています。フィックスパック 1 が別の代替インストール・パスにインストール済みの場合、フィックスパック 1 代替インストール・パスから以下のコマンドを呼び出すと、 DAS は、フィックスパック 1 代替インストール・パスから実行されるフィックスパック 1 に更新されます。

```
dasupdt -D
```

関連概念:

- 45 ページの『DB2 Administration Server』
- 58 ページの『Windows での DAS のセキュリティーに関する考慮事項』

DAS の除去

手順:

DAS を除去するには、以下のようにします。

- Windows オペレーティング・システムでは、次のようにします。
  1. サービスを除去するための正しい権限を持つアカウントまたはユーザー ID を使ってマシンにログオンします。
  2. **db2admin stop** を使用して DAS を停止します。
  3. sqllib サブディレクトリー下の db2das00 サブディレクトリーにあるすべてのファイル (必要に応じて) をバックアップします。

     注: この例では、除去する DAS の名前を db2das00 とします。ユーザーが、DB2DAS00 という名前の DB2 Universal Database™ (DB2 UDB) インスタンスを作成した場合、 DAS の名前を DB2DAS00 以外にすることも可能です。この場合、DAS の名前は DB2DAS01 (その名前も作成済みなら、DB2DAS02 以下に続きます) になります。接頭部 ″DB2DAS″ を持つサービスを検索し、存在する可能性のあるいくつかの DAS のリストから特定の DAS を識別することが必要です。 **db2admin** コマンドを何も指定せずに使用すれば、すべての DAS をリストできます。
  4. **db2admin drop** を使用して DAS をドロップします。
- UNIX オペレーティング・システムでは、次のようにします。
  1. DASADM 権限をもつユーザーとしてログインします。
  2. 次のいずれか 1 つを使用してスクリプトのセットアップを実行します。

     ```
     . DASHOME/das/dasprofile    (Bourne または Korn シェルの場合)
     source DASHOME/das/dascshrc (C シェルの場合)
     ```

     DASHOME は、DAS 所有者のホーム・ディレクトリーです。
  3. 次のように **db2admin** コマンドを使って DAS を停止します。

     ```
     db2admin stop
     ```
  4. DAS のホーム・ディレクトリー下の das サブディレクトリーにあるすべてのファイル (必要に応じて) をバックアップします。
  5. ログオフします。
  6. root としてログインし、次のように **dasdrop** コマンドを使って DAS を除去します。

dasdrop

dasdrop コマンドは、インストールした DB2 UDB バージョンおよびリリースに固有のサブディレクトリー下の instance サブディレクトリーにあります。

注: dasdrop コマンドを実行すると、 DB2 Administration Server (DAS) のホーム・ディレクトリーにある das サブディレクトリーが除去されます。

関連資料:

- 「コマンド・リファレンス」の『db2admin - DB2 Administration Server コマンド』
- 「コマンド・リファレンス」の『dasdrop - DB2 Administration Server の除去コマンド』

Enterprise Server Edition (ESE) システムでの DAS の設定

手順:

以下の情報は、DB2 Universal Database™ (DB2 UDB) ESE サーバー (Linux、Solaris オペレーティング環境、 Windows NT、 Windows 2000、 Windows Server 2003、 HP-UX、および AIX) をコントロール・センターを使用するリモート管理に構成するのに必要なステップを示します。

インストール中、セットアップ・プログラムはインスタンス所有マシン上に単一の DAS を作成します。他のマシン上に追加の DAS を作成して、コントロール・センターまたは構成アシスタントから、他のコーディネーター・ノードにアクセスすることが必要です。そうすれば、管理コーディネーター・ノードとして作業する場合のオーバーヘッドを、インスタンス内の複数区画に分散できます。インストール・プログラムは、そのプログラムが実行されるすべてのノードで DAS を作成します。 **db2setup** を使用しない場合のみ、これを手動で行う必要があります。

ここで説明する指示は、分割された ESE 環境にのみ適用します。単一区画 ESE システムだけで実行している場合は、この指示はご使用の環境には適用しません。

コーディネーター機能を分散するには、次のようにします。

1. パーティション・データベース・システム内で選択した追加マシンに新しい DAS を作成します。
2. コントロール・センターまたは構成アシスタント内で個々の DAS を別個のシステムとしてカタログします。
3. 個々の新しいシステム下の同じインスタンスをカタログし、その都度、DAS をカタログするために使用した同じマシン名を指定します。

構成には 2 つの局面があります。つまり、DB2 Administration Server (DAS) で必要なことと、ターゲットである管理される DB2 UDB インスタンスで推奨されていることです。以下の 3 つのセクションのうち、 2 つの構成トピックの説明にそれぞれ 1 つのセクションが割り振られています。 2 つの構成トピックの説明の前に、前提となる環境について説明しているセクションがあります。

環境の例

製品/バージョン:
DB2 UDB ESE V8.1

インストール・パス:
install_path

TCP サービス・ファイル:
サービス

DB2 UDB インスタンス:

名前: db2inst

所有者 ID:
db2inst

インスタンス・パス:
instance_path

ノード:
3 つのノード、db2nodes.cfg:

- 0 hostA 0 hostAswitch
- 1 hostA 1 hostAswitch
- 2 hostB 0 hostBswitch

DB 名:
db2instDB

DAS:

名前: db2as00

所有者/ユーザー ID:
db2as

インスタンス・パス:
das_path

インストール/実行ホスト:
hostA

ノード間通信ポート:
16000 (hostA および hostB 用の未使用ポート)

注: 上記のフィールドをサイト特有の値で置き換えてください。たとえば、次の表にはサポートされているいくつかの ESE プラットフォーム用のパス名の例が含まれています。

表 2. サポートされている ESE プラットフォーム用のパス名の例

パス	DB2 UDB ESE (AIX 版)	Solaris オペレーティング環境用の DB2 UDB ESE	DB2 UDB ESE (Windows 版)
install_path	/usr/opt/<v_r_ID>	/opt/IBM/db2/<v_r_ID>	C:¥sqllib
instance_path	/home/db2inst/sqllib	/home/db2inst/sqllib	C:¥profiles¥db2inst
das_path	/home/db2as/das	/home/db2as/das	C:¥profiles¥db2as
tcp_services_file	/etc/services	/etc/services	C:¥winnt¥system32¥drivers ¥etc¥services

この表で、<v_r_ID> はプラットフォーム固有のバージョンとリリースの ID です。たとえば、DB2 UDB (AIX 版) ESE バージョン 8 の場合は、 <v_r_ID> は db2_08_01 になります。

DB2 UDB ESE のインストール時に、セットアップ・プログラムはインスタンス所有マシン上に DAS を作成します。データベース・パーティション・サーバーは DAS と同じマシン上に常駐し、インスタンス用の接続点となります。つまり、このデータベース・パーティション・サーバーは、コントロール・センターまたは構成アシスタントからインスタンスに対して発行される要求のコーディネーター・ノードです。

DAS が各物理マシンごとにインストールされている場合、各マシンがコーディネーター・ノードとして動作します。各物理マシンは、コントロール・センターまたは構成アシスタント内で個々の DB2SYSTEM として表示されます。異なるクライアントが異なるシステムを使ってパーティション・データベース・サーバーに接続する場合、サーバーがコーディネーター・ノードの機能性を分散し、着信接続のバランスを取るのに役立ちます。

関連概念:

- 45 ページの『DB2 Administration Server』
- 63 ページの『Enterprise Server Edition (ESE) システムでの DAS 構成』

Enterprise Server Edition (ESE) システムでの DAS 構成

DAS は、ツールの代わりに特定のタスクを実行する管理コントロール・ポイントです。物理的なマシン 1 台につき設けることのできる DAS は多くて 1 つです。いくつかのマシンで構成される ESE インスタンスの場合は、コントロール・センターが ESE インスタンスを実行できるよう、すべてのマシンで DAS を実行している必要があります。この DAS (db2as) は、ターゲット DB2® Universal Database (DB2 UDB) インスタンス (db2inst) の親としてコントロール・センターのナビゲーター・ツリーに存在するシステムによって表されています。

たとえば、 db2inst は 2 つの物理マシンまたはホストにわたって分散している 3 つのノードから構成されています。 hostA および hostB で **db2as** を実行することによって、最小要件を満たすことができます。

注:

1. hostA に存在するパーティションの数は、hostA で実行できる DAS の数とは何の関係もありません。ホストに複数の論理ノード (MLN) 構成が存在するとしても、 hostA で実行できる DAS のコピーは 1 つのみです。
2. **dascrt** コマンドを使って個々に作成する必要のある、各マシン、または物理ノードで必要な DAS は 1 つです。各マシンまたは物理ノード上の DAS は、タスク・センターおよびコントロール・センターが正しく作動するように、稼働していなければなりません。 ID db2as は、hostA および hostB に存在する必要があります。 db2as ID のホーム・ディレクトリーは、2 つのシステム間でクロスマウントしてはいけません。その代わりに、DAS を hostA と hostB に作成するために、異なるユーザー ID を使用することができます。

DB2 Universal Database™ (DB2 UDB) Enterprise Server Edition for Windows® では、構成アシスタントまたはコントロール・センターを使って DB2 UDB サーバーへの接続構成を自動化する場合、 DAS として同一マシン上にあるデータベース・パーティション・サーバーはコーディネーター・ノードになります。つまり、クライアントからデータベースへの物理接続はすべて、他のデータベース・パーティション・サーバーに経路指定される前に、コーディネーター・ノードに向けられます。

DB2 Universal Database (DB2 UDB) Enterprise Server Edition for Windows では、他のマシン上で DB2 Administration Server を追加すると、構成アシスタントまたはコントロール・センターで DB2 ディスカバリーを使って他のシステムをコーディネーター・ノードとして構成することができます。

DB2 Universal Database (DB2 UDB) Enterprise Server Edition for Windows で作業する場合、リモート・コマンド・サービス (**db2rcmd.exe**) は、ノード内管理通信を自動的に処理します。

コントロール・センターは、TCP サービス・ポート 523 を使用して DAS と通信します。このポートは、DB2 UDB 専用として予約済みです。このため、TCP サービス・ファイルに新しい項目を挿入する必要はありません。

関連タスク:

- 47 ページの『DB2 Administration Server の作成』

関連資料:

- 「コマンド・リファレンス」の『db2admin - DB2 Administration Server コマンド』

Administration Server、インスタンス、およびデータベースのディスカバリー

リモート・マシンへの接続を構成する方法は 2 つあります。構成アシスタント内に構築される、ディスカバリー・サービスを使用するか、または Lightweight Directory Access Protocol (LDAP) などの既存のディレクトリー・サービスを使用するかのどちらかです。

このディスカバリー・サービスは、構成アシスタントおよび DB2® Administration Server に統合されています。リモート・マシンへの接続を構成するためには、ユーザーはクライアント・マシンにログオンし、構成アシスタント (CA) を実行します。 CA は、ネットワーク上のすべてのマシンにブロードキャスト・シグナルを送信します。 DAS をインストールし、ディスカバリー用に構成してあるマシンでは、そのマシン上のすべてのインスタンスとデータベース情報を含むパッケージを返信することにより、 CA からのブロードキャスト・シグナルに応答します。その後 CA は、このパッケージ中の情報を使ってクライアント接続を構成します。このディスカバリー方式を使用すると、リモート・サーバーのカタログ情報が、ローカル・データベースおよびノード・ディレクトリーで自動的に生成されます。

ディスカバリー方式では、すべてのクライアント・マシンにログオンし、 CA を実行することが必要です。クライアントが多数ある環境では、これは非常に難しく、時間がかかります。その場合、代替方法として、LDAP のようなディレクトリー・サービスを使用します。

既知ディスカバリー (Known Discovery) を使用すると、クライアントに認識されているシステム上のインスタンスとデータベースを検出することができます。また、新しいシステムを追加すれば、そのインスタンスとデータベースを検出することができます。検索ディスカバリー (Search Discovery) を使用すると、既知ディスカバリーの全機能が提供され、他の DB2 Universal Database™ (DB2 UDB) サーバーのためのローカル・ネットワークを検索できるオプションが追加されます。

システムで既知ディスカバリーをサポートするには、 DAS 構成ファイル中の *discover* パラメーターを KNOWN に設定します。システムで既知ディスカバリーと検索ディスカバリーの両方をサポートするには、 DAS 構成ファイル中の *discover* パラメーターを SEARCH に設定します (これがデフォルトです)。システムとそのインスタンスおよびデータベースのディスカバリーを行わないようにするには、このパラメーターを DISABLE に設定します。 DAS 構成ファイルで *discover* パラメーターを DISABLE に設定すると、システムのディスカバリーを行いません。

注: 検索ディスカバリーによってクライアントに戻される TCP/IP ホスト名は、**hostname** コマンドの入力時に DB2 UDB サーバー・システムによって戻されるのと同じホスト名です。クライアント側では、このホスト名によってマップされる IP アドレスは、クライアント・マシンで構成される TCP/IP ドメイン・ネーム・サーバー (DNS) か、あるいは、DNS が構成されていない場合はクライアントの *hosts* ファイル中のマッピング項目によって判別されます。 DB2 UDB サーバー・システムに複数のアダプター・カードを構成してある場合は、TCP/IP が正確なホスト名を戻すようにサーバー上で構成されており、 DNS またはローカル・クライアントの *hosts* ファイルが任意の IP アドレスにホスト名をマップすることを確かめます。

クライアント側では、*discover* パラメーターを使ってディスカバリー機能も使用可能にします。ただし、この場合の *discover* パラメーターは次のように、クライアント・インスタンス (またはクライアントとして動作するサーバー) で設定します。

- **KNOWN**

  既知ディスカバリーは構成アシスタントおよびコントロール・センターによって使用され、すでにローカル・システムが認識しているシステムに関連するインスタンス、およびデータベース情報を検索します。新しいシステムは、ツールで提供される**「システムの追加」**機能を使って追加できます。 *discover* パラメーターを KNOWN に設定すると、ネットワークを検索できなくなります。

- **SEARCH**

  既知ディスカバリーの全機能を使用可能にし、ローカル・ネットワークを検索することができます。つまり、すべての検索がローカル・ネットワークに限定されるということです。

  「他のシステム (ネットワークの検索) (Other Systems (Search the network))」アイコンは、この選択項目を選択した場合だけ表示されます。これがデフォルト設定です。

- **DISABLE**

  ディスカバリーを使用不能にします。この例では、「データベースの追加ウィザード (Add Database Wizard)」で、**「ネットワークの検索 (Search the network)」**オプションを使用することができません。

注: *discover* パラメーターはデフォルトにより、すべてのクライアントおよびサーバー・インスタンスに SEARCH が設定されます。 *discover* パラメーターはデフォルトにより、すべての DB2 Administration Server (DAS) に SEARCH が設定されます。

関連概念:

- 79 ページの『Lightweight Directory Access Protocol (LDAP) ディレクトリー・サービス』

関連タスク:

- 66 ページの『ディスカバリーからサーバー・インスタンスおよびデータベースを隠す』
- 67 ページの『ディスカバリー・パラメーターの設定』

ディスカバリーからサーバー・インスタンスおよびデータベースを隠す

手順:

サーバー・システム上に複数のインスタンスがあり、それらのインスタンス内に複数のデータベースがあるかもしれません。そのようなインスタンスやデータベースをディスカバリー・プロセスから隠したいと思うことがあります。

クライアントがシステム上のサーバー・インスタンスを検出できるようにするには、システム上の各サーバー・インスタンスの *discover_inst* データベース・マネージャー構成パラメーターを ENABLE (デフォルト値) に設定します。このパラメーターを DISABLE に設定すると、このインスタンスとデータベースをディスカバリーから隠すことができます。

クライアントからデータベースを検出できるようにするには、 *discover_db* データベース構成パラメーターを ENABLE (デフォルト値) に設定します。このパラメーターを DISABLE に設定すると、データベースをディスカバリーから隠すことができます。

注: インスタンスを検出したい場合、DAS 構成ファイルで、 *discover* も KNOWN または SEARCH に設定する必要があります。データベースを検出したい場合は、サーバー・インスタンスで、 *discover_inst* パラメーターを使用可能にすることも必要です。

関連資料:

- 「*管理ガイド: パフォーマンス*」の『discover_inst - 「サーバー・インスタンスの検出」構成パラメーター』
- 「*管理ガイド: パフォーマンス*」の『discover_db - 「データベース・ディスカバリー」構成パラメーター』

ディスカバリー・パラメーターの設定

手順:

discover パラメーターは、サーバー・システム上の DAS 構成ファイル、およびクライアント上のデータベース・マネージャー構成ファイルで設定されます。構成アシスタントまたはコントロール・センターは、データベース・マネージャー構成パラメーター *discover*、*discover_inst*、 *discover_db* を設定するのに使用します。パラメーターの設定方法は、次のとおりです。

- DAS 上で以下のようにします。

  次のコマンド・プロセスを使って DAS 構成ファイル中の *discover* パラメーターを (例として) 更新します。

  ```
  update admin cfg using discover [ DISABLE | KNOWN |
     SEARCH ]
  ```

  DAS *discover* 構成パラメーターは、オンラインで構成可能です。つまり、変更を有効にするために DAS を停止してから再始動する必要はありません。

  注: 検索ディスカバリーは TCP/IP でのみ動作します。

- 構成アシスタントで以下のように作業します。

  コマンド行 (すべてのプラットフォーム) で **db2ca** と入力することによって、または「スタート」メニューから (Windows の場合)、**「スタート」—>「プログラム」—>「IBM DB2」—> 「セットアップ・ツール (Set-up Tools)」—> 「構成アシスタント (Configuration Assistant)」**をクリックすることによって、構成アシスタントを開始します。

  構成アシスタントを使用してデータベース・マネージャー構成パラメーターを設定するには、次のようにします。

1. **「構成」—>「DBM 構成」**をクリックします。
2. 変更するキーワードをクリックします。
3. 値の列で、変更するキーワードの値をクリックし、**「OK」**をクリックします。
4. もう一度**「OK」**をクリックします。すると、メッセージが表示されます。**「閉じる (Close)」**をクリックします。

コントロール・センターを使って、 *discover_inst* および *discover_db* パラメーターを設定します。

また、構成アシスタントを使って構成パラメーターを更新することもできます。

関連資料:

- 「*管理ガイド: パフォーマンス*」の『discover_inst - 「サーバー・インスタンスの検出」構成パラメーター』
- 「*管理ガイド: パフォーマンス*」の『discover_db - 「データベース・ディスカバリー」構成パラメーター』

- 「コマンド・リファレンス」の『UPDATE ADMIN CONFIGURATION コマンド』
- 「*管理ガイド: パフォーマンス*」の『discover - 「DAS ディスカバリー・モード」構成パラメーター』

構成アシスタントおよびコントロール・センターを使用するための DAS の設定

前提条件:

ネットワーク上のシステムについての情報を検索するには、**discover** を構成しなければなりません。

制約事項:

DAS は、各物理区画上になければなりません。 DAS が区画上に作成されると、DB2SYSTEM 名は TCP/IP ホスト名に構成され、 **discover** 設定のデフォルトが SEARCH になります。

手順:

DB2 ディスカバリーは、構成アシスタントとコントロール・センターが使用する機能です。この機能の構成は、 DB2 Administration Server (DAS) 構成とインスタンスのデータベース・マネージャー構成を更新して、 DB2 ディスカバリーが正確な情報を検索することを確認する場合に必要です。

クライアントが構成アシスタントからディスカバリー要求を出す場合、ディスカバリーを使用可能にした各 DAS が応答します。パーティション・データベース環境では、各物理区画が個別の DB2SYSTEM 名として応答します。管理可能な実際のインスタンスは、その物理区画が認識するインスタンスに依存します。インスタンスが複数の区画に広がる場合があるので、同じインスタンスが潜在的に、異なるシステム名を介して管理されることがあります。この機能を使うと、サーバー・インスタンスでロードのバランスを取るのに役立ちます。たとえば、インスタンス ″A″ がシステム ″S1″ およびシステム ″S2″ を介して使用可能である場合、何人かのユーザーが ″S1″ を使ってデータベースのカタログを作成し、別のユーザーが ″S2″ を使って同じデータベースのカタログを作成できます。各ユーザーが、異なるコーディネーター・データベース・パーティションを使ってサーバーに接続できます。

関連資料:

- 「コマンド・リファレンス」の『db2ilist - インスタンスのリスト・コマンド』
- 「コマンド・リファレンス」の『db2ncrt - インスタンスへのデータベース・パーティション・サーバーの追加コマンド』
- 「*管理ガイド: パフォーマンス*」の『discover - 「DAS ディスカバリー・モード」構成パラメーター』

ディスカバリー用の DAS 構成の更新

制約事項:

DAS は、各物理区画上になければなりません。 DAS が区画上に作成されると、DB2SYSTEM 名は TCP/IP ホスト名に構成され、 *discover* 設定のデフォルトが SEARCH になります。

手順:

ディスカバリーによって検索されるシステム名は、 DB2 Administration Server (DAS) が常駐するシステムです。接続が確立されると、ディスカバリーは検索したシステムをコーディネーター・ノードとして使用します。

DAS 構成の更新時、および DB2 Universal Database™ (DB2 UDB) システムからコーディネーター・ノードを選択できるようにするには、個々の DB2 Administration Server の構成ファイルで *discover*=SEARCH (デフォルト値) を設定します。

パーティション・データベース・サーバー環境に複数の DAS があると、構成アシスタントまたはコントロール・センターのインターフェース上の複数のシステムに同じインスタンスが表示される場合がありますが、各システムにはインスタンスへの異なる通信アクセス・パスが指定されます。ユーザーは、通信用のコーディネーター・ノードとして異なる DB2 UDB システムを選択して、ワークロードを再配分することができます。

関連資料:

- 「*管理ガイド: パフォーマンス*」の『その他の変数』

DB2 Administration Server での First Failure Data Capture (FFDC)

First Failure Data Capture (FFDC) は、DB2® Administration Server が、エラー発生時に自動的にキャプチャーする一連の診断情報に適用される一般用語です。この情報により、診断情報を得るためにエラーを再現する必要がなくなります。診断情報は、一か所にまとめられています。

DB2 Administration Server FFDC が捕そくする情報には、次のものがあります。

- 管理通知ログ。

  イベントの発生時、DB2 Administration Server は情報を DB2 Administration Server ログ・ファイル db2dasdiag.log に書き込みます。
- ダンプ・ファイル。

  エラーの状態によっては、失敗したプロセス ID の名前が付いた外部バイナリー・ダンプ・ファイルに追加情報が記録されます。これらのファイルは、DB2 Universal Database™ (DB2 UDB) お客様サポートが使用するためのものです。
- トラップ・ファイル。

  DB2 Administration Server は、トラップやセグメンテーション違反、例外などにより処理が続行できない時、トラップ・ファイルを生成します。トラップ・ファイルには、問題発生前に実行された最後のステップの関数の流れが含まれています。

DB2 Administration Server での First Failure Data Capture (FFDC) は、情報のロケーションをキャプチャーします。

デフォルトでは、DB2 Administration Server FFDC 情報は次のロケーションにあります。

- Windows® システムの場合:

  DB2INSTPROF 環境変数が設定されていない場合は以下のとおりです。

  ```
  db2path¥DB2DAS00¥dump
  ```

  db2path は DB2PATH 環境変数で参照されるパスで、 DB2DAS00 は DAS サービスの名前です。 DAS 名は、 **db2admin** コマンドを引き数なしで入力することによって取得できます。

  DB2INSTPROF 環境変数が設定されている場合は以下のとおりです。

  ```
  x:¥db2instprof¥DB2DAS00¥dump
  ```

  x: は DB2PATH 環境変数で参照されるドライブ、 db2instprof はインスタンス・プロファイル・ディレクトリー、 DB2DAS00 は DAS サービスの名前です。

- UNIX® ベースのシステムでは、次のようにします。

  ```
  $DASHOME/das/dump
  ```

  $DASHOME は、DAS ユーザーのホーム・ディレクトリーです。

注: ダンプ・ディレクトリーを周期的にクリーンアップして、大きくなりすぎないようにしてください。

DB2 Administration Server ログの解釈

DB2 Administration Server ログ・ファイル (db2dasdiag.log) のフォーマットは、DB2 FFDC ログ・ファイル db2diag.log のフォーマットと類似しています。db2dasdiag.log ファイルの解釈方法については、トラブルシューティングのトピックにある、管理ログの解釈のセクションを参照してください。

関連概念:

- 45 ページの『DB2 Administration Server』

第 3 章 データベースの作成

この章では、データベース設計のインプリメンテーションを構成する、さまざまなオブジェクトを個々に簡潔に説明しています。

前の章では、データベースを作成する前に知っている必要がある情報を中心に説明しました。また、いくつかのトピックや、データベースの作成前に済ませておくべきタスクも記載されていました。

この部の最後の章では、データベースを変更する前に考慮しなければならないトピックが示されています。また、データベース・オブジェクトの変更方法やドロップ方法について説明されています。

データベースの作成

前提条件:

作成前に、データベースの内容、レイアウト、潜在的な増大度、および使用方法の設計に十分な時間をかけてください。

手順:

データベースの作成時には、以下のタスクがそれぞれ実行されます。

- データベースで必要になるすべてのシステム・カタログ表を設定する。
- データベースのリカバリー・ログを割り当てる。
- データベースの構成ファイルを作成し、デフォルト値を設定する。
- データベースのユーティリティーをデータベースにバインドする。

システム・カタログ・ビュー上の CREATETAB、BINDADD、CONNECT、IMPLICIT_SCHEMA、および SELECT の各データベース特権は、PUBLIC に付与されます。

コントロール・センターを使用してデータベースを作成するには、以下のようにします。

1. オブジェクト・ツリーを順に展開し、「**データベース (Databases)**」フォルダーを表示します。
2. 「**データベース (Databases)**」フォルダーを右マウス・ボタン・クリックし、ポップアップ・メニューから**「作成 (Create)」—>「標準」** または**「作成 (Create)」—> 「自動保守」** を選択します。
3. ステップに従ってこのタスクを完了します。

以下のコマンド行プロセッサー・コマンドは、デフォルトの位置に `personl` と呼ばれるデータベースを、 ″Personnel DB for BSchiefer Co″ という関連する注釈を付けて作成します。

```
CREATE DATABASE personl
   WITH "Personnel DB for BSchiefer Co"
```

データベースの作成時、構成パラメーターすべてのデフォルト値を受け入れないで、構成アドバイザーを使用し、データベースの構成を援助するように要求することもできます。これは、**CREATE DATABASE** コマンドで AUTOCONFIGURE オプションを使用することによって実行できます。

```
CREATE DATABASE <database name>
   AUTOCONFIGURE
```

AUTOCONFIGURE 文節にはいくつかのオプションがあります。区画に分割された環境でデータベースを作成する場合、AUTOCONFIGURE 文節は使用できません。

データベースが作成されると同時に、詳細デッドロック・イベント・モニターも作成されます。他のモニターと同様に、このイベント・モニターにも関連したオーバーヘッドがあります。詳細デッドロック・イベント・モニターを必要としない場合は、以下のコマンドを使用してイベント・モニターをドロップできます。

```
DROP EVENT MONITOR db2detaildeadlock
```

このイベント・モニターが消費するディスク・スペースの量を制限するために、出力ファイルの最大数に達すると、イベント・モニターが非アクティブになり、メッセージが管理通知ログに書き込まれます。必要のない出力ファイルを除去すると、イベント・モニターは次のデータベースの活動化時にアクティブになります。

データベースを別の (通常はリモートの) データベース・マネージャー・インスタンスに作成することができます。このタイプの環境では、デフォルト・インスタンス以外のインスタンス (リモート・インスタンスを含む) に対してインスタンス・レベルの管理を実行することができます。

関連概念:

- 「*管理ガイド: プランニング*」の『データベースに記録される事柄』
- 6 ページの『データベース・マネージャーの複数インスタンス』
- 252 ページの『データベース権限』
- 「*管理ガイド: プランニング*」の『追加のデータベース設計に関する考慮事項』

関連資料:

- 「*コマンド・リファレンス*」の『CREATE DATABASE コマンド』

初期データベース・パーティション・グループの定義

データベースを最初に作成するときに、 db2nodes.cfg ファイルに指定されたすべての区画について、データベース・パーティションが作成されます。その他の区画は、**ADD DBPARTITIONNUM** および **DROP DBPARTITIONNUM VERIFY** コマンドを使用して追加またはドロップすることができます。

以下の 3 つのデータベース・パーティション・グループが定義されます。

- SYSCATSPACE 表スペース用の IBMCATGROUP (システム・カタログ表を保持します)

- TEMPSPACE1 表スペース用の IBMTEMPGROUP (データベース処理の間に作成された一時表を保持します)
- USERSPACE1 表スペース用の IBMDEFAULTGROUP (デフォルトで、ユーザー表と索引を保持します)

関連概念:

- 「*管理ガイド: プランニング*」の『データベース・パーティション・グループ』

関連資料:

- 「*コマンド・リファレンス*」の『ADD DBPARTITIONNUM コマンド』
- 「*コマンド・リファレンス*」の『DROP DBPARTITIONNUM VERIFY コマンド』

初期の表スペースの定義

データベースを作成するとき、以下の 3 つの表スペースが定義されます。

- システム・カタログ表用の SYSCATSPACE
- データベース処理中に作成されたシステム一時表用の TEMPSPACE1
- ユーザー定義の表および索引用の USERSPACE1

注: データベースを初めて作成する時点では、 USER TEMPORARY 表スペースは作成されません。

CREATE DATABASE コマンドを使用してどの表スペース・パラメーターも指定していない場合は、システム管理ストレージ (SMS) のディレクトリー・コンテナーを使用して、データベース・マネージャーがこれらの表スペースを作成します。ディレクトリー・コンテナーはデータベースのサブディレクトリーに作成されます。これらの表スペースのエクステント・サイズはデフォルトに設定されます。

前提条件:

データベースが作成されること、および表スペースを作成する権限が必要です。

手順:

コントロール・センターを使用して初期表スペースを定義するには、以下のようにします。

1. オブジェクト・ツリーを順に展開し、**「データベース (Databases)」**フォルダーを表示します。
2. 「**データベース (Databases)**」フォルダーを右マウス・ボタン・クリックし、ポップアップ・メニューから**「作成 (Create)」—>「標準」** または**「作成 (Create)」—> 「自動保守」** を選択します。
3. ステップに従ってこのタスクを完了します。

コマンド行を使用して初期表スペースを定義するには、以下のように入力します。

```
CREATE DATABASE <name>
  CATALOG TABLESPACE
    MANAGED BY SYSTEM USING ('<path>')
```

```
  EXTENTSIZE <value> PREFETCHSIZE <value>
USER TABLESPACE
  MANAGED BY DATABASE USING (FILE'<path>' 5000,
                             FILE'<path>' 5000)
  EXTENTSIZE <value> PREFETCHSIZE <value>
TEMPORARY TABLESPACE
  MANAGED BY SYSTEM USING ('<path>')
WITH "<comment>"
```

表スペースにデフォルト定義を使用したくない場合は、 **CREATE DATABASE** コマンドでこれらの特性を指定することができます。たとえば、次のコマンドは Windows 上にデータベースを作成するために使用するものです。

```
CREATE DATABASE PERSONL
  CATALOG TABLESPACE
    MANAGED BY SYSTEM USING ('d:¥pcatalog','e:¥pcatalog')
    EXTENTSIZE 16 PREFETCHSIZE 32
  USER TABLESPACE
    MANAGED BY DATABASE USING (FILE'd:¥db2data¥personl' 5000,
                               FILE'd:¥db2data¥personl' 5000)
    EXTENTSIZE 32 PREFETCHSIZE 64
  TEMPORARY TABLESPACE
    MANAGED BY SYSTEM USING ('f:¥db2temp¥personl')
   WITH "Personnel DB for BSchiefer Co"
```

この例では、最初の表スペースのそれぞれの定義が明示的に提供されています。デフォルト定義を使用したくない表スペースの表スペース定義を指定するだけで済みます。

注: パーティション・データベース環境で作業している場合、特定のパーティションに対して、コンテナーを作成したり割り当てたりすることはできません。まずデフォルト・ユーザーおよび一時表スペースで、データベースを作成する必要があります。次に、CREATE TABLESPACE ステートメントを使用して、必要な表スペースを作成できます。最後に、デフォルトの表スペースをドロップします。

CREATE DATABASE コマンドの MANAGED BY 句をコーディングした後に、CREATE TABLESPACE ステートメントの MANAGED BY 句と同じ形式が続きます。

関連概念:

- 75 ページの『システム・カタログ表の定義』
- 「*管理ガイド: プランニング*」の『表スペースの設計』

関連タスク:

- 85 ページの『表スペースの作成』

関連資料:

- 「コマンド・リファレンス」の『CREATE DATABASE コマンド』

バッファー・プールの作成

データベース・マネージャーで使用するために、新規のバッファー・プールを作成できます。バッファー・プールは、データベース・システムのパフォーマンスを即時に向上させます。

バッファー・プールに選択するページ・サイズは、表スペースに指定されたページ・サイズによって決定されます。バッファー・プールに使用するページ・サイズは、バッファー・プールを作成した後ではもはや変更できないため、このページ・サイズの選択は重要です。

前提条件:

このステートメントの許可 ID には、SYSCTRL 権限または SYSADM 権限がなければなりません。

新規のバッファー・プールを作成する前に、以下の点を解決してください。

- 使用するバッファー・プールの名前
- バッファー・プールを即時に作成するのか、あるいは次にデータベースが非活動化および再活動化されるときに続けて作成するのか
- バッファー・プールを、データベースを構成するすべてのデータベース・パーティションのサブセットに関連付けるかどうか
- ページ・サイズや、ページ数に基づくバッファー・プールの合計サイズを含め、バッファー・プールのサイズを制御するパラメーターに関連付ける値
- 拡張ストレージを使用するか、ブロック・ベースのサポートを使用するか、あるいはそのどちらも使用しないか

手順:

新規バッファー・プールを作成するには、次のようにします。

1. `SELECT BPNAME FROM SYSCAT.BUFFERPOOLS` を実行し、データベース内に既に存在するバッファー・プール名をリストします。
2. 結果として表示されたリストに現在存在していないバッファー・プール名を選択します。名前の先頭に 『SYS』 または 『IBM』 の文字を使用することはできません。
3. 作成しようとしているバッファー・プールの特性を決定します。
4. CREATE BUFFERPOOL ステートメントを実行する適正な許可 ID があることを確認します。
5. CREATE BUFFERPOOL ステートメントを実行します。

関連タスク:

- 181 ページの『バッファー・プールの変更』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『CREATE BUFFERPOOL ステートメント』

システム・カタログ表の定義

1 組のシステム・カタログ表が、それぞれのデータベースごとに作成され維持されます。これらの表には、データベース・オブジェクト (たとえば、表、ビュー、索引、およびパッケージ) の定義についての情報と、これらのオブジェクトに対してユーザーが持っているアクセスのタイプについてのセキュリティーの情報が含まれています。これらの表は、SYSCATSPACE 表スペースに保管されます。

これらの表は、表作成時など、データベースの操作中に更新されます。これらの表を明示的に作成したりドロップしたりすることはできませんが、内容の照会や表示は可能です。データベースが作成されると、システム・カタログ表オブジェクトに加えて、次のデータベース・オブジェクトがシステム・カタログで定義されます。

- スキーマ SYSIBM、SYSFUN、および SYSPROC 中の一連のルーチン (関数およびプロシージャー)。
- システム・カタログ表の一連の読み取り専用のビューが SYSCAT スキーマに作成されます。
- 一連の更新可能なカタログ・ビューが SYSSTAT スキーマに作成されます。更新可能なビューを使用すると、特定の統計情報を使用して、仮定データベースのパフォーマンスを調査したり、 **RUNSTATS** ユーティリティーを使用しないで統計を更新したりすることができます。

データベースが作成された後で、システム・カタログ・ビューのアクセスを制限することができます。

関連概念:

- 「*SQL リファレンス 第 1 巻*」の『ユーザー定義関数』
- 「*SQL リファレンス 第 1 巻*」の『カタログ・ビュー』
- 「*SQL リファレンス 第 1 巻*」の『関数の概要』

関連タスク:

- 276 ページの『システム・カタログ・ビューのセキュリティー』

関連資料:

- 「*SQL リファレンス 第 1 巻*」の『関数』

データベース・ディレクトリーの定義

以下の 3 つのディレクトリーが、新しいデータベースの確立またはセットアップのときに使用されます。

- ローカル・データベース・ディレクトリー
- システム・データベース・ディレクトリー
- ノード・ディレクトリー

ローカル・データベース・ディレクトリー

データベースが定義されているパス (Windows® オペレーティング・システムでは「ドライブ」)ごとに、*ローカル・データベース・ディレクトリー・ファイル*が 1 つずつあります。このディレクトリーには、そこからアクセスできるデータベースごとに 1 つの項目が入っています。各項目には次のものが含まれています。

- **CREATE DATABASE** コマンドによって提供されたデータベース名
- データベースの別名 (別名が指定されない場合は、データベース名と同じ)
- データベースを説明する注釈 (**CREATE DATABASE** コマンドで提供されたもの)
- データベースのためのルート・ディレクトリーの名前
- その他のシステム情報

関連資料:

- 「コマンド・リファレンス」の『CREATE DATABASE コマンド』

システム・データベース・ディレクトリー

システム・データベース・ディレクトリー・ファイルは、データベース・マネージャーの各インスタンスごとに存在し、このインスタンスについてカタログされているそれぞれのデータベースごとに 1 つの項目が含まれています。データベースは **CREATE DATABASE** コマンドの発行時に暗黙のうちにカタログされますが、**CATALOG DATABASE** コマンドによって明示的にカタログすることもできます。

作成されたそれぞれのデータベースごとに 1 つの項目がディレクトリーに追加されますが、これには以下の情報が含まれます。

- **CREATE DATABASE コマンド**によって提供されたデータベース名
- データベースの別名 (別名が指定されない場合は、データベース名と同じ)
- **CREATE DATABASE** コマンドによって提供されたデータベース注釈
- ローカル・データベース・ディレクトリーの位置
- データベースが*間接* データベースであることを示す標識。これは、データベースが現行のデータベース・マネージャー・インスタンス上にあるという意味です。
- その他のシステム情報

UNIX® プラットフォームおよびパーティション・データベース環境では、すべてのデータベース・パーティションが、同じシステム・データベース・ディレクトリー・ファイル (そのインスタンスのホーム・ディレクトリーの `sqldbdir` サブディレクトリーの中にある `sqldbdir`) を常にアクセスするようにしなければなりません。同じ `sqldbdir` サブディレクトリーの中の、システム・データベース・ディレクトリーまたはシステム・インテンション・ファイル `sqldbins` のいずれかが、共用ファイル・システム上にある別のファイルに対する記号リンクである場合、予期しないエラーが発生する可能性があります。

関連タスク:

- 13 ページの『データベースでのデータ・パーティションを使用可能にする』
- 83 ページの『データベースのカタログ作成』

関連資料:

- 「コマンド・リファレンス」の『CREATE DATABASE コマンド』

データベースの代替サーバーを識別する

サーバー・クラッシュが発生すると、そのサーバーに接続している各クライアントは通信エラーを受け取り、接続が終了してアプリケーション・エラーが発生します。可用性を保つ必要が大きい場合には、通常、重複セットアップ、またはサーバーをスタンバイ・ノードにフェイルオーバーする機能をインプリメントします。どちらの場合も、DB2 Universal Database® (DB2 UDB) クライアント・コードは、フェイルオーバー・ノードで稼働している可能性のある元のサーバーとの接続を再確立するか (IP アドレスもまたフェイルオーバーします)、新しいサーバーとの接続を再確立しようとします。

手順:

新規の代替サーバーを定義するには、 UPDATE ALTERNATE SERVER FOR DATABASE コマンドを使用します。このコマンドは、システム・データベース・ディレクトリー内で、データベース別名の代替サーバー情報を更新します。

関連概念:

- 81 ページの『自動クライアント・リルートのインプリメンテーション』
- 331 ページの『クライアントの自動転送についての説明およびセットアップ』

ローカルまたはシステムのデータベース・ディレクトリー・ファイルの表示

システム上のデータベースに関連した情報を参照する必要があります。

前提条件:

ローカルまたはシステム・データベース・ディレクトリー・ファイルのどちらかを見る前に、インスタンスおよびデータベースを前もって作成しておく必要があります。

手順:

ローカル・データベース・ディレクトリー・ファイルの内容を見るためには、以下のコマンドを出します (ただし、<location> はデータベースの位置を指定します)。

```
LIST DATABASE DIRECTORY ON <location>
```

システム・データベース・ディレクトリー・ファイルの内容を見るためには、データベース・ディレクトリー・ファイルの位置を*指定せずに* **LIST DATABASE DIRECTORY** コマンドを出します。

関連資料:

- 「コマンド・リファレンス」の『LIST DATABASE DIRECTORY コマンド』

ノード・ディレクトリー

データベース・マネージャーは、最初のデータベース・パーティションがカタログされるときに*ノード・ディレクトリー* を作成します。データベース・パーティションをカタログするためには、**CATALOG NODE** コマンドを使用します。ローカルのノード・ディレクトリーの内容をリストするには、 **LIST NODE DIRECTORY** コマンドを使用します。ノード・ディレクトリーは、各データベース・クライアントごとに作成され維持されます。ディレクトリーには、そのクライアントがアクセスできる 1 つ以上のデータベースを持っている各リモート・ワークステーションごとに 1 つの項目が入っています。 DB2® クライアントは、データベース接続またはインスタンス接続が要求されると、ノード・ディレクトリーの中の通信エンドポイント情報を使用します。

ディレクトリーの中の項目には、クライアントからリモート・データベース・パーティションに通信するために使用される、通信プロトコルのタイプについての情報

も含まれます。ローカル・データベース・パーティションをカタログすることによって、同じマシン上に常駐するインスタンスに対する別名が作成されます。

関連資料:

- 「コマンド・リファレンス」の『CATALOG TCPIP NODE コマンド』
- 「コマンド・リファレンス」の『LIST NODE DIRECTORY コマンド』
- 「コマンド・リファレンス」の『CATALOG NETBIOS NODE コマンド』
- 「コマンド・リファレンス」の『CATALOG LOCAL NODE コマンド』
- 「コマンド・リファレンス」の『CATALOG NAMED PIPE NODE コマンド』

Lightweight Directory Access Protocol (LDAP) ディレクトリー・サービス

ディレクトリー・サービスとは、分散環境内にある複数のシステムおよびサービスについてのリソース情報を集めたリポジトリーです。クライアントとサーバーはディレクトリー・サービスを使用して、それらのリソースにアクセスします。クライアントおよびサーバーは、ディレクトリー・サービスを使用して、他のリソースにアクセスする方法を見つけます。分散環境にある、これら他のリソースについての情報を、ディレクトリー・サービス・リポジトリーに入力することが必要です。

Lightweight Directory Access Protocol (LDAP) は、ディレクトリー・サービスに対する業界標準のアクセス方式です。各データベース・サーバーのインスタンスは自らの存在を LDAP サーバーに公表するとともに、データベースの作成時にはデータベース情報を LDAP ディレクトリーへ送信します。クライアントがデータベースに接続すると、 LDAP ディレクトリーからそのサーバーのカタログ情報を取り出せます。各クライアントは、それぞれのマシンでローカルにカタログ情報を保管する必要はなくなります。クライアント・アプリケーションは、LDAP ディレクトリーの中で、データベースへ接続するのに必要な情報を探します。

DB2® UDB システムの管理者として、ディレクトリー・サービスを確立および保守できます。構成アシスタントまたはコントロール・センターは、このディレクトリー・サービスの保守を支援できます。ディレクトリー・サービスは、Lightweight Directory Access Protocol (LDAP) ディレクトリー・サービスを通して DB2 UDB に対して使用可能になります。 LDAP ディレクトリー・サービスを使用するには、まず DB2 がサポートする LDAP サーバーが存在しており、ディレクトリー情報がそこに保管されるようにする必要があります。

注: Windows® 2000 ドメイン環境で実行中の場合、LDAP サーバーは、 Windows 2000 アクティブ・ディレクトリーに統合されているので、すでに使用可能になっています。その結果、Windows 2000 を実行するすべてのマシンで LDAP を使用できます。

LDAP ディレクトリーは、クライアントが非常に多いために各クライアント・マシンでローカル・ディレクトリー・カタログを更新するのが困難なエンタープライズ環境で役立ちます。この場合、LDAP サーバーにディレクトリー項目を保管し、カタログ項目の保守が 1 か所、つまり LDAP サーバーで実行されるようにすること

をお勧めします。 LDAP サーバーの購入および保守にはかなりコストがかかることが考えられるので、クライアントの数がコストに見合う十分なものである場合のみ、考慮してください。

関連概念:

- 64 ページの『Administration Server、インスタンス、およびデータベースのディスカバリー』
- 337 ページの『Lightweight Directory Access Protocol (LDAP) の紹介』

データベース・パーティション・グループ (nodegroups) の作成

CREATE DATABASE PARTITION GROUP ステートメントを使用してデータベース・パーティション・グループを作成します。このステートメントは、表スペース・コンテナーおよび表データが常駐するデータベース・パーティションのセットを指定します。このステートメントは、以下のことも行います。

- データベース・パーティション・グループのパーティション・マップの作成。
- パーティション・マップ ID の生成。
- 以下のカタログ表へのレコードの挿入。
  - SYSCAT.DBPARTITIONGROUPS
  - SYSCAT.PARTITIONMAPS
  - SYSCAT.DBPARTITIONGROUPDEF

前提条件:

マシンおよびシステムが使用可能で、かつパーティション・データベース環境を扱えなければなりません。 DB2 Universal Database Enterprise - Server Edition が購入され、インストール済みであるはずです。データベースが存在していることが必要です。

手順:

コントロール・センターを使用してデータベース・パーティションを作成するには、以下のようにします。

1. オブジェクト・ツリーを順に展開し、**「データベース・パーティション・グループ (Database partition groups)」**フォルダーを表示します。
2. **「データベース・パーティション・グループ (Database partition groups)」** フォルダーを右クリックして、ポップアップ・メニューから**「作成 (Create)」**を選択します。
3. 「データベース・パーティション・グループの作成 (Create Database partition groups)」ウィンドウで、情報をすべて入力し、矢印を使用してノードを**「使用可能ノード (Available nodes)」**ボックスから**「選択済みデータベース・パーティション (Selected database partitions)」** ボックスに移動して、**「OK」**をクリックします。

コマンド行を使用してデータベース・パーティション・グループを作成するには、以下のように入力します。

```
CREATE DATABASE PARTITION GROUP <name> ON PARTITIONS (<value>,<value>)
```

データベース内のデータベース・パーティションのサブセット上にいくつかの表をロードするとします。以下のコマンドを使用して、少なくとも 3 つのデータベー

ス・パーティション (0 ～ 2) からなるデータベース内に、 2 つのデータベース・パーティション (1 と 2) のデータベース・パーティション・グループを作成します。

```
CREATE DATABASE PARTITION GROUP mixng12 ON PARTITIONS (1,2)
```

CREATE DATABASE コマンドまたは sqlecrea() API は、デフォルトのシステム・データベース・パーティション・グループである、IBMDEFAULTGROUP、IBMCATGROUP、および IBMTEMPGROUP も作成します。

関連概念:

- 「*管理ガイド: プランニング*」の『データベース・パーティション・グループ』
- 「*管理ガイド: プランニング*」の『パーティション・マップ』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『CREATE DATABASE PARTITION GROUP ステートメント』
- 「*管理 API リファレンス*」の『sqlecrea - データベースの作成』
- 「*コマンド・リファレンス*」の『CREATE DATABASE コマンド』

データベース・リカバリー・ログの定義

データベース・リカバリー・ログ は、新しい表の追加または既存の表に対する更新を含む、データベースに対して行われたすべての変更の記録を保持します。このログはいくつかの*ログ・エクステント* からなり、それぞれのログ・エクステントは、*ログ・ファイル* と呼ばれる別個のファイルに入っています。

データベース・リカバリー・ログを使用して、障害 (たとえば、システム電源異常またはアプリケーション・エラー) によって、データベースが矛盾した状態のままにならないようにすることができます。障害が発生した場合、すでに入力はされたがコミットされていない変更事項はロールバックされ、コミット済みのすべてのトランザクション (ディスクに物理的に書き込まれてはいないかもしれない) は再実行されます。これらのアクションによって、データベースの保全性が保たれます。

関連概念:

- 「*データ・リカバリーと高可用性 ガイドおよびリファレンス*」の『リカバリー・ログについて』

自動クライアント・リルートのインプリメンテーション

DB2® Universal Database (DB2 UDB) クライアント・アプリケーションが DB2 UDB サーバーとの通信を失った場合に、管理者の介入なしにクライアントをその状態からリカバリーすることができます。 DB2 UDB は、DB2 UDB サーバーへのクライアント通信のリカバリーをサポートしています。通信障害が発生する前に、クライアント接続が認識する代替ロケーションを設定しておく必要があります。

UPDATE ALTERNATE SERVER FOR DATABASE コマンドを使用して、特定のデータベース上に代替サーバー・ロケーションを定義します。代替ホスト名およびポ

ート番号は、コマンドの一部として指定します。このロケーションは、サーバーのシステム・データベース・ディレクトリーに保管されます。

サーバー・インスタンスの特定のデータベース上にある代替サーバー・ロケーションを指定した後、接続プロセスの一環として、代替サーバー・ロケーション情報がクライアントに戻されます。何らかの理由でクライアントとサーバーとの間の通信が失われた場合、コード化された DB2 UDB クライアントは代替サーバー情報を使用して接続の再確立を試行します。 DB2 UDB クライアントは、元のサーバーと代替サーバーとに対して交互に接続を試行します。これらの試行のタイミングは、最初は迅速に試行を繰り返し、徐々に試行と試行の間のインターバルが長くなる仕方で変化します。

接続が成功すると、通信障害の後にデータベース接続が再確立されたことを示す SQLCODE -30108 が戻されます。ホスト名 / IP アドレス、およびサービス名 / ポート番号が戻されます。クライアント通信の再確立が元のサーバーに対しても代替サーバーに対しても不可能な場合、クライアント・コードは元の通信障害に関するエラーだけをアプリケーションに戻します。

関連概念:

- 331 ページの『クライアントの自動転送についての説明およびセットアップ』
- 332 ページの『クライアント自動転送の制約事項』

関連資料:

- 334 ページの『クライアント自動転送の例』

データベースへのユーティリティーのバインド

データベースの作成時に、データベース・マネージャーによって db2ubind.lst 内のユーティリティーをデータベースにバインドすることが試みられます。このファイルは、sqllib ディレクトリーの bnd サブディレクトリーに格納されています。

ユーティリティーをバインドすると、*パッケージ* が作成されます。これは、1 つのソース・ファイルからの特定の SQL ステートメントを処理するのに必要な情報がすべて入れられているオブジェクトです。

注: クライアントからこれらのユーティリティーを使用する場合は、ユーティリティーを明示的にバインドする必要があります。

何らかの理由で、データベースにユーティリティーをバインドまたは再バインドする必要がある場合、コマンド行プロセッサーを使用して、以下のコマンドを出します。

```
connect to sample
bind @db2ubind.lst
```

注: sample データベースにパッケージを作成するには、これらのファイルが入っているディレクトリーを使用しなければなりません。バインド・ファイルは、sqllib ディレクトリーの bnd サブディレクトリーの中にあります。この例では、sample がデータベースの名前です。

関連タスク:

- 71 ページの『データベースの作成』

関連資料:

- 「コマンド・リファレンス」の『BIND コマンド』

データベースのカタログ作成

新しいデータベースを作成すると、システム・データベース・ディレクトリーのファイルに自動的にカタログされます。また、**CATALOG DATABASE** コマンドを使って、システム・データベース・ディレクトリーのファイルにデータベースを明示的にカタログすることもできます。 **CATALOG DATABASE** コマンドを使えば、違う別名でデータベースをカタログしたり、 **UNCATALOG DATABASE** コマンドによって以前に削除したデータベース項目をカタログしたりすることが可能になります。

前提条件:

データベースは、データベース作成時に自動的にカタログされますが、それでもデータベースをカタログすることが必要になる場合があります。これを実行する場合、データベースが存在していることが必要です。

手順:

次のコマンド行プロセッサー・コマンドを使うと、 personl データベースが humanres としてカタログされます。

```
CATALOG DATABASE personl AS humanres
   WITH "Human Resources Database"
```

この場合、システム・データベース・ディレクトリー項目のデータベース別名は humanres になります。これは、データベース名 (personl) とは違うものです。

デフォルト以外のインスタンスにデータベースをカタログすることもできます。次の例では、データベース B への接続は、 INSTNC_C に対して行われます。インスタンス instnc_c を、このコマンド試行前に、ローカル・ノードとしてカタログしておくことが必要です。

```
CATALOG DATABASE b as b_on_ic AT NODE instnc_c
```

注: クライアント・ノードで **CATALOG DATABASE** コマンドを使うと、データベース・サーバー・マシンにあるデータベースをカタログすることもできます。

注: デフォルトで、データベース・ディレクトリーを含むディレクトリー・ファイルは、「ディレクトリー・キャッシュ・サポート (*dir_cache*)」構成パラメーターを使ってメモリーのキャッシュに入れられます。ディレクトリー・キャッシュが使用可能な場合は、別のアプリケーションがディレクトリーを変更しても (たとえば、 **CATALOG DATABASE** または **UNCATALOG DATABASE** コマンドを使用して)、ユーザーのアプリケーションが再始動されるまで、その変更は有効になりません。コマンド行プロセッサー・セッションが使用するディレクトリー・キャッシュをリフレッシュするには、 **db2 terminate** コマンドを出してください。

パーティション・データベースでは、ディレクトリー・ファイルのキャッシュは各データベース・パーティションで作成されます。

アプリケーション・レベルのキャッシュに加えて、データベース・マネージャー・レベルのキャッシュも、内部的なデータベース・マネージャーの索引に使用されます。この「共用」キャッシュをリフレッシュするには、 **db2stop** および **db2start** コマンドを出してください。

関連タスク:

- 84 ページの『リモート・データベース・サーバー・マシンの情報を使用したディレクトリーの更新』

関連資料:

- 「*管理ガイド: パフォーマンス*」の『dir_cache - 「ディレクトリー・キャッシュ・サポート」構成パラメーター』
- 「コマンド・リファレンス」の『CATALOG DATABASE コマンド』
- 「コマンド・リファレンス」の『TERMINATE コマンド』
- 「コマンド・リファレンス」の『UNCATALOG DATABASE コマンド』

リモート・データベース・サーバー・マシンの情報を使用したディレクトリーの更新

手順:

構成アシスタント (CA)・インタープリターのデータベース追加ウィザードを使って、カタログ項目を作成することができます。 DB2 Application Development Client がある場合、アプリケーション・プログラムを作成して項目をカタログすることができます。

注: データベースでカタログを作成するには、SYSADM または SYSCTRL 権限が必要です。または、*catalog_noauth* 構成パラメーターを YES に設定する必要があります。

コマンド行プロセッサーを使用してディレクトリーを更新するには、以下のことを行います。

1. 次のコマンドの 1 つを使って、ノード・ディレクトリーを更新します。
   - APPC 接続があるノードの場合:

     ```
     db2 CATALOG APPC NODE <nodename>
       REMOTE <symbolic_destination_name> SECURITY <security_type>
     ```

     たとえば、次のようになります。

     ```
     db2 CATALOG APPC NODE DB2NODE REMOTE DB2CPIC SECURITY PROGRAM
     ```

   - TCP/IP 接続を持つ DB2 Universal Database for z/OS and OS/390 バージョン 5.1 以降または DB2 Universal Database for AS/400 バージョン 4.2 以降のデータベースの場合:

     ```
     db2 CATALOG TCPIP NODE <nodename>
        REMOTE <hostname> or <IP address>
        SERVER <service_name> or <port_number>
        SECURITY <security_type>
     ```

たとえば、次のようになります。

```
db2 CATALOG TCPIP NODE MVSIPNOD REMOTE MVSHOST SERVER DB2INSTC
```

DB2 for OS/390 and z/OS の TCP/IP 接続に使用されるデフォルト・ポートは 446 です。

2. DB2 Connect で作業している場合、CATALOG DCS DATABASE コマンドを使用して DCS ディレクトリーを更新することを考慮する必要があります。

リモート・クライアントがある場合、各リモート・クライアントでもディレクトリーを更新することが必要です。

関連概念:

- 「*DB2 Connect* ユーザーズ・ガイド」の『システム・データベース・ディレクトリーの値』
- 「*DB2 Connect* ユーザーズ・ガイド」の『DCS ディレクトリーの値』

関連資料:

- 「コマンド・リファレンス」の『CATALOG DATABASE コマンド』
- 「コマンド・リファレンス」の『CATALOG DCS DATABASE コマンド』
- 「コマンド・リファレンス」の『CATALOG APPC NODE コマンド』
- 「コマンド・リファレンス」の『CATALOG TCPIP NODE コマンド』
- 「コマンド・リファレンス」の『CATALOG NETBIOS NODE コマンド』
- 「コマンド・リファレンス」の『CATALOG APPN NODE コマンド』

表スペースの作成

表スペースは、データベース・システムによって使用される物理ストレージ・デバイスと、データの保管に使用される論理コンテナーまたは表との関係を確立します。

前提条件:

表スペース作成時に参照する、コンテナーのデバイス名またはファイル名を知っておくことが必要です。加えて、表スペースに割り振られる各デバイスまたはファイル名と関連するスペースも知っておくべきです。

手順:

データベースの中に表スペースを作成すると、その表スペースにコンテナーが割り当てられ、その定義と属性がデータベース・システム・カタログに記録されます。このようにして、その表スペースに表を作成できるようになります。

コントロール・センターを使用して表スペースを作成するには、以下のようにします。

1. オブジェクト・ツリーを順に展開し、**「表スペース (Table spaces)」**フォルダーを表示します。
2. **「表スペース (Table spaces)」**フォルダーを右クリックして、ポップアップ・メニューから**「作成 (Create)」—>「ウィザードを使用する表スペース (Table Spaces Using Wizard)」**を選択します。
3. このウィザードのステップに従って、タスクをすべて実行します。

コマンド行を使用して SMS 表スペースを作成するには、以下のように入力します。

```
CREATE TABLESPACE <NAME>
   MANAGED BY SYSTEM
   USING ('<path>')
```

コマンド行を使用して DMS 表スペースを作成するには、以下のように入力します。

```
CREATE TABLESPACE <NAME>
   MANAGED BY DATABASE
   USING (FILE'<path>' <size>)
```

次の SQL ステートメントは、別個の 3 つのドライブの 3 つのディレクトリーを使用して、 Windows 上で SMS 表スペースを作成するものです。

```
CREATE TABLESPACE RESOURCE
   MANAGED BY SYSTEM
   USING ('d:¥acc_tbsp', 'e:¥acc_tbsp', 'f:¥acc_tbsp')
```

次の SQL ステートメントは、それぞれ 5,000 ページの各ファイル・コンテナーを使用して、 DMS 表スペースを作成します。

```
CREATE TABLESPACE RESOURCE
   MANAGED BY DATABASE
   USING (FILE'd:¥db2data¥acc_tbsp' 5000,
          FILE'e:¥db2data¥acc_tbsp' 5000)
```

上記の 2 つの例では、コンテナーに明示的な名前が提供されました。しかし、相対コンテナー名を指定する場合、コンテナーはデータベース用に作成されたサブディレクトリーの中に作成されます。

さらに、指定されたパス名の一部が存在しない場合は、データベース・マネージャーが作成します。サブディレクトリーがデータベース・マネージャーに作成される場合は、表スペースがドロップされると、そのサブディレクトリーもデータベース・マネージャーによって削除されます。

上記の例は、表スペースが特定のデータベース・パーティション・グループに関連付けられていないことを前提としています。デフォルトのデータベース・パーティション・グループである IBMDEFAULTGROUP は、以下のパラメーターがステートメント内に指定されていない場合に使用されます。

```
IN database_partition_group_name
```

以下の SQL ステートメントは、それぞれ 10 000 ページの 3 つの論理ボリュームを使用して UNIX ベースのシステム上に DMS 表スペースを作成し、それらの入出力特性を指定します。

```
CREATE TABLESPACE RESOURCE
   MANAGED BY DATABASE
   USING (DEVICE '/dev/rdblv6' 10000,
          DEVICE '/dev/rdblv7' 10000,
          DEVICE '/dev/rdblv8' 10000)
   OVERHEAD 12.67
   TRANSFERRATE 0.18
```

この SQL ステートメントで指定している UNIX デバイスは、すでに存在しているものであり、インスタンス所有者および SYSADM グループが書き込みを行えるようでなければなりません。

以下の例は、UNIX パーティション・データベース内の ODDGROUP と呼ばれるデータベース・パーティション・グループ上に DMS 表スペースを作成します。ODDGROUP は、 CREATE DATABASE PARTITION GROUP ステートメントを使用してあらかじめ作成されていなければなりません。この場合、ODDGROUP データベース・パーティション・グループは、1、3、および 5 の番号が付いたデータベース・パーティションから成っていると想定されています。すべてのデータベース・パーティション上で、10 000 の 4 KB ページについて、デバイス /dev/hdisk0 を使用します。さらに、40 000 の 4 KB ページから成るデータベース・パーティションごとに 1 つのデバイスを宣言します。

```
CREATE TABLESPACE PLANS IN ODDGROUP
   MANAGED BY DATABASE
   USING (DEVICE '/dev/HDISK0' 10000, DEVICE '/dev/n1hd01' 40000)
          ON DBPARTITIONNUM 1
         (DEVICE '/dev/HDISK0' 10000, DEVICE '/dev/n3hd03' 40000)
          ON DBPARTITIONNUM 3
         (DEVICE '/dev/HDISK0' 10000, DEVICE '/dev/n5hd05' 40000)
          ON DBPARTITIONNUM 5
```

UNIX デバイスは次の 2 つの区分に分類されます。それは、文字シリアル・デバイスとブロック構造デバイスです。すべてのファイル・システム・デバイスの場合、各ブロック・デバイス (つまり*クックド・デバイス*) ごとに、対応する文字シリアル・デバイス (つまり*ロー・デバイス*) を持っているのが普通です。ブロック構造デバイスは、普通、″hd0″ または ″fd0″ のような名前で指定されます。文字シリアル・デバイスは、普通、″rhd0″、″rfd0″、または ″rmt0″ のような名前で指定されます。これらの文字シリアル・デバイスは、ブロック・デバイスよりも速いアクセス速度を持っています。文字シリアル・デバイス名は、CREATE TABLESPACE コマンド上で使用する必要があり、ブロック・デバイス名は使用できません。

オーバーヘッドと転送速度は、 SQL ステートメントのコンパイル時に使用する最善のアクセス・パスを決定するための目安になります。現行のデフォルトは、以下のとおりです。

- OVERHEAD 12.67 ms
- TRANSFERRATE 0.18 ms

DB2 UDB で順次プリフェッチ機能を使用すると、並列入出力が使われ、順次入出力のパフォーマンスを大幅に向上させることができます。

デフォルトの 4 KB サイズより大きいページ・サイズを使用する表スペースを作成することもできます。次の SQL ステートメントは、UNIX ベースのシステムで 8 KB の SMS 表スペースを作成するものです。

```
CREATE TABLESPACE SMS8K
   PAGESIZE 8192
   MANAGED BY SYSTEM
   USING ('FSMS_8K_1')
   BUFFERPOOL BUFFPOOL8K
```

関連したバッファー・プールのページ・サイズも、同じ 8 KB でなければならないことに注意してください。

作成された表スペースは、それが参照するバッファー・プールが活動化されるまで使用できません。

ALTER TABLESPACE SQL ステートメントを使うと、 DMS 表スペースでコンテナーを追加、ドロップ、またはサイズ変更したり、表スペースの PREFETCHSIZE、OVERHEAD、および TRANSFERRATE の設定値を変更したりすることができます。表スペース・ステートメントを発行するトランザクションは、システム・カタログの競合を避けるために、できるだけ早くコミットする必要があります。

注: PREFETCHSIZE は、EXTENTSIZE の整数倍でなければなりません。たとえば、EXTENTSIZE が 10 なら PREFETCHSIZE は 20 や 30 などにする必要があります。表スペースを作成するときには、以下の式を使用してプリフェッチ・サイズを手動で作成してください。

プリフェッチ・サイズ = (コンテナーの数) X (コンテナーごとの物理スピンドルの数) X エクステント・サイズ

また、DB2 UDB がプリフェッチ・サイズを自動的に決めるようにすることも検討してください。

関連概念:

- 「*管理ガイド: プランニング*」の『表スペースの設計』
- 「*管理ガイド: プランニング*」の『システム管理スペース』
- 「*管理ガイド: プランニング*」の『データベース管理スペース』
- 「*管理ガイド: パフォーマンス*」の『順次プリフェッチ』

関連タスク:

- 「*管理ガイド: プランニング*」の『64 ビット環境におけるラージ・ページのサポートの使用可能化 (AIX)』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『ALTER TABLESPACE ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『CREATE TABLESPACE ステートメント』

特殊タイプの表スペースの作成

データベース・マネージャー、およびアプリケーションとユーザーが使用する、さまざまな表スペースのタイプがあります。

SYSTEM TEMPORARY 表スペースの作成

SYSTEM TEMPORARY 表スペースはデータベースの作成時にデフォルトで作成されますが、システム・ソート・タスクを処理するための個別の表スペースを割り当てたほうがよいかもしれません。

前提条件:

SYSTEM TEMPORARY 表スペースに関連したコンテナーがなければなりません。

制約事項:

システム一時表を格納できるのは SYSTEM TEMPORARY 表スペースだけなので、データベースは常に少なくとも 1 つの SYSTEM TEMPORARY 表スペースを持っていなければなりません。

手順:

SYSTEM TEMPORARY 表スペースは、システム 一時表を格納するのに使用されます。データベースを作成すると、定義される 3 つのデフォルト表スペースのうち 1 つが、 ″TEMPSPACE1″ という SYSTEM TEMPORARY 表スペースになります。

CREATE TABLESPACE ステートメントを使用して、別の SYSTEM TEMPORARY 表スペースを作成することもできます。たとえば、

```
CREATE SYSTEM TEMPORARY TABLESPACE tmp_tbsp
   MANAGED BY SYSTEM
   USING ('d:¥tmp_tbsp','e:¥tmp_tbsp')
```

各ページ・サイズの表スペースが少なくとも 1 つ必要です。

SYSTEM TEMPORARY 表スペースの作成時に指定できるデータベース・パーティション・グループは、 IBMTEMPGROUP だけです。

関連タスク:

- 89 ページの『USER TEMPORARY 表スペースの作成』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『CREATE TABLESPACE ステートメント』

USER TEMPORARY 表スペースの作成

USER TEMPORARY 表スペースは、デフォルトではデータベース作成の時点で作成されません。データベース内のデータを扱うアプリケーション・プログラムの作業の一部として、一時表を使用することが必要な場合があります。そのような場合は、ユーザー一時表を作成する必要があります。

手順:

USER TEMPORARY 表スペースは、宣言された一時表を格納するのに使用されます。

CREATE TABLESPACE ステートメントを使用して、 USER TEMPORARY 表スペースを作成できます。

```
CREATE USER TEMPORARY TABLESPACE usr_tbsp
   MANAGED BY DATABASE
   USING (FILE 'd:¥db2data¥user_tbsp' 5000,
          FILE 'e:¥db2data¥user_tbsp' 5000)
```

REGULAR 表スペースと同様、USER TEMPORARY 表スペースは、IBMTEMPGROUP 以外の任意のデータベース・パーティション・グループに作成できます。 USER TEMPORARY 表スペースの作成時には、 IBMDEFAULTGROUP がデフォルトのデータベース・パーティション・グループとして使用されます。

DECLARE GLOBAL TEMPORARY TABLE ステートメントは、宣言された一時表を、USER TEMPORARY 表スペース内で使用できるように定義します。

関連タスク:

- 109 ページの『ユーザー定義の一時表の作成』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『CREATE TABLESPACE ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『DECLARE GLOBAL TEMPORARY TABLE ステートメント』

データベース・パーティション・グループでの表スペースの作成

複数パーティションを持つデータベース・パーティション・グループの中に表スペースを置くことによって、その表スペース内のすべての表が、そのデータベース・パーティション・グループ内の各パーティションにわたって分割、つまりパーティション化されます。表スペースはデータベース・パーティション・グループ内に作成されます。いったん 1 つのデータベース・パーティション・グループ内に入ると、表スペースは、そこにとどまらなければならず、別のデータベース・パーティション・グループに変更することはできません。 CREATE TABLESPACE ステートメントは、表スペースとデータベース・パーティション・グループを関連付けるために使用されます。

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『CREATE TABLESPACE ステートメント』

ロー I/O の指定

データを保管するためにコンテナーを処理している場合、 DB2 Universal Database は、直接ディスク・アクセス (ロー I/O) をサポートしています。このタイプのサポートにより、直接ディスク・アクセス (ロー) デバイスを DB2 Universal Database システムにアタッチすることができます。 (唯一の例外は、Windows 9x オペレーティング・システムです。)

前提条件:

表スペース作成時に参照する、コンテナーのデバイス名またはファイル名を知っておくことが必要です。表スペースに割り振られる各デバイスまたはファイル名と関連するスペースの量も知っておくべきです。

コンテナーに対する読み取りおよび書き込みの正しい許可が必要です。

手順:

以下のリストでは、直接ディスクにアクセスするタイプのデバイスを識別するための物理的また論理的方法が示されています。

- Windows で物理ハード・ディスクを指定するには、次の構文を使用します。

  ¥¥.¥PhysicalDriveN

  ここで N は、システムにある物理ドライブのいずれかを表します。この場合、N を 0、1、2、または他の正の整数に置き換えることができます。

  ¥¥.¥PhysicalDrive5
- Windows で論理ドライブ (つまり未フォーマットのパーティション) を指定するには、次の構文を使用します。

  ¥¥.¥N:

  ここで N: は、システムにある論理ドライブ名を表します。たとえば、N: を E: または他のドライブ名で置き換えることができます。ドライブを識別するための文字の使用によって課せられる制限をなくすには、論理ドライブでグローバル・ユニーク ID (GUID) を使用できます。
- **注:** デバイスに書き込みを行うには、 Windows NT バージョン 4.0 (サービス・パック 3 以降を適用済み) をインストールしておく必要があります。
- UNIX ベースのプラットフォームでは、論理ボリュームを、単一で連続する拡張可能なディスク・ボリュームとしてユーザーおよびアプリケーションに示すことができます。そのように示しても、論理ボリュームを、不連続の物理区画や、複数の物理ボリュームに置くことができます。また、論理ボリュームは、単一のボリューム・グループ内に入っていなければなりません。 1 つのボリューム・グループにつき 256 の論理ボリュームという制限があります。 1 つのボリューム・グループあたりの物理ボリュームの制限は 32 です。 **mklv** コマンドを使用して、追加の論理ボリュームを作成することができます。このコマンドにより、論理ボリュームの名前を指定し、その論理ボリュームのために割り振られる論理区画の数やロケーションを含む、論理ボリュームの特性を定義することができます。

  論理ボリュームを作成した後で、**chlv** コマンドでその名前や特性を変更することができ、 **extendlv** でその論理ボリュームに割り振られている論理区画の数を増やすことができます。さらに大きい数が指定されていない限り、作成の時点での論理ボリュームのデフォルトの最大サイズは 512 論理区画です。この制限をオーバーライドするには、**chlv** コマンドを使用します。

  AIX では、論理ボリューム・ストレージの確立と制御を可能にするオペレーティング・システム・コマンドのセット、ライブラリー・サブルーチン、および他のツールは、論理ボリューム・マネージャー (LVM) と呼ばれています。 LVM

は、ストレージ・スペースのより単純で柔軟な論理ビューと実際の物理的なディスクとの間のデータをマップすることにより、ディスク・リソースを制御します。

mklv および他の論理ボリューム・コマンド、また LVM の詳細については、「*AIX 5L バージョン 5.2 システム・マネージメント・コンセプト:オペレーティング・システムおよびデバイス*」を参照してください。

Windows 2000 以降の場合、DMS 未加工表スペース・コンテナーの指定の方式は新しくなっています。ボリューム (つまり基本ディスク・パーティションまたは動的ボリューム) には、その作成時にグローバル・ユニーク ID (GUID) が割り当てられます。 GUID は、表スペース定義でコンテナーを指定する際に、デバイス ID として使用できます。 GUID はすべてのシステムでユニークです。つまり、複数パーティション・データベース構成では、ディスク・パーティション定義が同じであっても、GUID は各パーティションで異なるということです。

Windows システムで定義されるすべてのディスク・ボリュームで GUID を表示しやすくするため、 *db2listvolumes.exe* というツールを使用できます (Windows オペレーティング・システムのみ)。このツールは、ツールが実行する現行ディレクトリーで 2 つのファイルを作成します。 1 つは `volumes.xml` というファイルで、XML が使用できるブラウザーで、 XML でエンコードされた各ディスク・ボリュームを簡単に表示するための情報が入っています。 2 つ目のファイルは `tablespace.ddl` で、表スペース・コンテナーを指定するために必要な構文が入っています。表スペース定義に必要な残りの情報が入るように、このファイルを更新する必要があります。 *db2listvolumes* ツールではコマンド行引き数は必要ありません。

関連タスク:

- 92 ページの『Linux でのロー I/O の設定』

Linux でのロー I/O の設定

データを保管するためにコンテナーを処理している場合、 DB2 Universal Database は、直接ディスク・アクセス (ロー I/O) をサポートしています。このタイプのサポートにより、直接ディスク・アクセス (ロー) デバイスを DB2 Universal Database システムにアタッチすることができます。 Linux 環境で作業中の場合、特定情報があります。

前提条件:

表スペース作成時に参照する、コンテナーのデバイス名またはファイル名を知っておくことが必要です。表スペースに割り振られる各デバイスまたはファイル名と関連するスペースも知っておくべきです。

Linux でロー I/O をセットアップするには、以下が必要になります。

- 1 つ以上の IDE または SCSI 空きディスク区分
- ロー・デバイス・コントローラー `/dev/rawctl` または `/dev/raw`。これらのコントローラーがない場合、次のようにしてシンボリック・リンクを作成してください。

```
# ln -s /dev/your_raw_dev_ctrl /dev/rawctl
```

- 通常、Linux 配布版で提供される raw ユーティリティー

注: ロー・デバイス・ノード名は、現在ロー入出力をサポートしている配布版によって異なります。

表 3. ロー I/O をサポートする Linux 配布版

配布版	ロー・デバイス・ノード	ロー・デバイス・コントローラー
RedHat または TurboLinux	/dev/raw/raw1 から 255	/dev/rawctl
SuSE	/dev/raw1 から 63	/dev/raw

DB2 は、これら 2 つのロー・デバイス・コントローラー、およびロー・デバイス・ノードのその他ほとんどの名前をサポートしています。ロー・デバイスは、Linux/390 での DB2 ではサポートされていません。

手順:

Linux には、ブロック・デバイスにバインドしなければロー入出力を実行できない、ロー・デバイス・ノードのプールがあります。ロー・デバイスをブロック・デバイスにバインドするための情報の中央リポジトリーとして機能するロー・デバイス・コントローラーがあります。バインドは、一般的に Linux ディストリビューターによって提供されているユーティリティー raw を使用して実行されます。

Linux でロー入出力を構成するには、次のようにしてください。

この例では、使用されるロー区分は /dev/sda5 です。この区分には、重要なデータは含まれていないと想定します。

ステップ 1. この区分の 4 096 バイト・ページの数を計算します。端数が出た場合は切り捨てます。たとえば、以下のとおりです。

```
# fdisk /dev/sda
Command (m for help): p

Disk /dev/sda: 255 heads, 63 sectors, 1106 cylinders
Units = cylinders of 16065 * 512 bytes
```

表 4. Linux ロー I/O 計算

デバイス・ブート	開始	終了	ブロック	ID	システム
/dev/sda1	1	523	4200997	83	Linux
/dev/sda2	524	1106	4682947+	5	Extended
/dev/sda5	524	1106	4682947	83	Linux

```
Command (m for help): q
#
```

/dev/sda5 のページ数:

```
num_pages = floor( ((1106-524+1)*16065*512)/4096 )
num_pages = 11170736
```

ステップ 2. 未使用のロー・デバイス・ノードをこの区分にバインドします。マシンをリブートするたびに行う必要があり、root アクセスが必要です。raw -a を使用して、どのロー・デバイス・ノードが使用されているかを調べます。

```
# raw /dev/raw/raw1 /dev/sda5
/dev/raw/raw1: bound to major 8, minor 5
```

ステップ 3. ロー・デバイス・コントローラーおよびディスク・パーティションに、適切な読み取りアクセス権を設定します。ロー・デバイスには、適切な読み取り/書き込みアクセス権を設定します。

ステップ 4. ディスク区分ではなくロー・デバイスを指定して、DB2 に表スペースを作成します。たとえば、以下のとおりです。

```
CREATE TABLESPACE dms1
MANAGED BY DATABASE
USING (DEVICE '/dev/raw/raw1' 11170736)
```

ロー・デバイスの表スペースは、DB2 でサポートされているその他すべてのページ・サイズでもサポートされています。

関連タスク:

- 90 ページの『ロー I/O の指定』

スキーマの作成

データを表に編成すれば、表や他の関連オブジェクトを一緒にグループ化する利点もあります。これは、CREATE SCHEMA ステートメントを使用して、スキーマを定義することによって行うことができます。スキーマについての情報は、接続するデータベースのシステム・カタログ表に保持されます。他のオブジェクトが作成されると、それらのオブジェクトをこのスキーマ内に置くことができます。

前提条件:

一緒にグループ化されるデータベース表および他の関連オブジェクトが存在していなければなりません。

制約事項:

このステートメントは、DBADM 権限を持つユーザーが発行する必要があります。

スキーマは、ユーザーが IMPLICIT_SCHEMA 権限を持ったときに、暗黙に作成されることもあります。この権限を使用して、ユーザーは、すでに存在していないスキーマ名を持つオブジェクトを作成するときに、常に暗黙にスキーマを作成します。

ユーザーが IMPLICIT_SCHEMA 権限を持っていない場合、ユーザー自身の許可 ID と同じ名前のスキーマのみを作成することができます。

スキーマはデータベース内での固有性を強制するために使用されるため、スキーマ内のオブジェクトに対する非修飾アクセスは許可されません。 2 人のユーザーが 2 つの表 (または他のオブジェクト) を同時に作成できるかどうかを考慮する際に、このことが明らかになります。スキーマを使用して固有性を強制しないと、 3 人目のユーザーが表を照会しようとする場合にあいまいさが生じます。もっと詳しく制限されていなければ、どの表を使用すべきかを判別できません。

新しいスキーマ名は、すでにシステム・カタログに存在する名前にすることはできず、 ″SYS″ で始めることはできません。

手順:

ユーザーが SYSADM 権限または DBADM 権限を持っている場合、そのユーザーは、任意の有効な名前でスキーマを作成することができます。データベースが作成されるときに、IMPLICIT_SCHEMA 権限が PUBLIC (つまり、すべてのユーザー) に付与されます。

CREATE SCHEMA ステートメントの一部として作成されたどのオブジェクトの定義者も、スキーマの所有者になります。この所有者は、他のユーザーに対して、スキーマ特権の GRANT および REVOKE を行うことができます。

表名に対する制限の一部としてスキーマ名を入力しなくても、別のユーザーが表にアクセスできるようにするには、そのユーザーについてビューを設定する必要があります。ビューの定義では、完全修飾表名 (ユーザーのスキーマを含む) を定義します。そうすれば、ユーザーはビュー名を使用して照会するのみで済みます。ビューは、ユーザーのスキーマによってビュー定義の一部として完全修飾されています。

コントロール・センターを使用してスキーマを作成するには、以下のようにします。

1. データベース内でオブジェクト・ツリーを順に展開し、**「スキーマ (Schema)」**フォルダーを表示します。
2. **「スキーマ (Schema)」**フォルダーを右クリックして、**「作成 (Create)」**をクリックします。
3. 新しいスキーマに関する情報をすべて入力し、**「OK」**をクリックします。

コマンド行を使用してスキーマを作成するには、以下のように入力します。

```
CREATE SCHEMA <name> AUTHORIZATION <name>
```

以下は、許可 ID ″joe″ を持つ個人ユーザーについてのスキーマを作成する、CREATE SCHEMA ステートメントの例です。

```
CREATE SCHEMA joeschma AUTHORIZATION joe
```

関連概念:

- 8 ページの『スキーマ別のオブジェクトのグループ化』
- 254 ページの『暗黙スキーマ権限 (IMPLICIT_SCHEMA) に関する考慮事項』
- 255 ページの『スキーマの特権』

関連タスク:

- 96 ページの『スキーマの設定』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『CREATE SCHEMA ステートメント』

スキーマの作成に関する詳細

スキーマは、データベース内でオブジェクト所有権を編成するのに使用されます。

スキーマの設定

いくつかのスキーマが存在している場合、特定の DB2 接続内から発行された動的 SQL ステートメント中の修飾されていないオブジェクト参照が使用できるよう、デフォルトとしてスキーマを設定するとよいでしょう。

手順:

デフォルト・スキーマの確立は、デフォルトとして使用したいスキーマに、特殊レジスター CURRENT SCHEMA を設定することによって行うことができます。この特殊レジスターは、どのユーザーでも設定することができます。設定するための許可は必要ありません。

以下に CURRENT SCHEMA 特殊レジスターの設定方法の例を示します。

```
SET CURRENT SCHEMA = 'SCHEMA01'
```

このステートメントは、アプリケーション・プログラム内から使用するか、あるいは対話式に発行することができます。一度設定されると、CURRENT SCHEMA 特殊レジスターの値は、データベース・オブジェクトへの修飾されていない参照が存在する CREATE SCHEMA ステートメント以外の動的 SQL ステートメント用の修飾子 (スキーマ) として使用されます。

CURRENT SCHEMA 特殊レジスターの初期値は、現行セッション・ユーザーの許可 ID と同じになります。

関連概念:

- 「*SQL リファレンス 第 1 巻*」の『スキーマ』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『SET SCHEMA ステートメント』
- 「*SQL リファレンス 第 1 巻*」の『予約済みスキーマ名と予約語』
- 「*SQL リファレンス 第 1 巻*」の『CURRENT SCHEMA 特殊レジスター』

第 4 章 表および関連する表オブジェクトの作成

この章では、データベース設計をインプリメントする際に、特定の特性を持つ表を作成する方法について説明します。

表の作成とデータの読み込み

表は、データベース内のデータのメイン・リポジトリーです。表の作成と、その表へのデータの入力は、新しいデータベースの作成時に行われます。

前提条件:

データを入れる表の設計と編成には時間をかけることが必要です。

手順:

表内でのデータの編成方法を決めたら、次の段階は、CREATE TABLE ステートメントを使って表を作成します。表の説明は、接続先のデータベースのシステム・カタログの中に格納されます。

CREATE TABLE ステートメントを使うと、表の名前として修飾付きまたは修飾なしの ID が付けられ、表の各列の定義が与えられます。表ごとに別々の表スペースに保管して、 1 つの表スペースには 1 つの表しか含まれないようにすることができます。表のドロップや作成を頻繁に行う場合は、表をそれぞれ別の表スペースに格納し、表ではなく表スペースをドロップするほうが効率的です。 1 つの表スペース内に多数の表を保管することもできます。パーティション・データベース環境では、選択された表スペースは、表データが保管されるデータベース・パーティション・グループおよびデータベース・パーティションも定義します。

表には、当初、何もデータが入っていません。表にデータ行を加えるには、次のどちらかを使用します。

- INSERT ステートメント
- LOAD または IMPORT コマンド
- パーティション・データベース環境で作業している場合は、オートローダー・ユーティリティー

変更内容をロギングに記録せずに、表にデータを追加することができます。CREATE TABLE ステートメントの NOT LOGGED INITIALLY 文節を使用すると、変更内容は表にロギングされません。表が作成されたのと同じ作業単位内の INSERT、DELETE、UPDATE、CREATE INDEX、DROP INDEX、または ALTER TABLE の操作によって表に対して行われた変更は、いずれもログに記録されません。ロギングは、後続の作業単位で開始します。

表には、1 つまたは複数の列定義が含まれています。 1 つの表について、最大 500 列を定義することができます。列は、エンティティーの属性を表します。 1 つの列の値は、すべて同じタイプの情報です。

注: 4 KB のページ・サイズを使用しているときには、最大で 500 列です。ページ・サイズが 8 KB、16 KB、または 32 KB のときは、最大で 1012 列です。

列定義には、*列名*、データ・タイプ、および必要に応じて *NULL 値属性* またはデフォルト値 (ユーザーがオプションで選択します) が含まれます。

列名は列に含まれる情報について記述したもので、簡単に識別できるものにすべきです。これはその表内ではユニークなものでなければなりませんが、他の表では同じ名前を使用することができます。

列のデータ・タイプは、列の値の長さと有効なデータの種類を示すものです。データベース・マネージャーで使うデータ・タイプには、文字ストリング、数値、日付、時間、ラージ・オブジェクトがあります。 GRAPHIC ストリング・データ・タイプは、マルチバイト文字セットを使用するデータベース環境のみで利用できます。また、ユーザー定義特殊タイプで列を定義することもできます。

デフォルト属性の指定は、値が指定されていない場合にどの値を使用するかを指定するものです。デフォルト値を指定するか、またはシステム定義のデフォルト値を使用することができます。デフォルト値は、NULL 値属性を指定した列でも指定しない列でも指定することができます。

NULL 値属性仕様は、列に NULL 値を含めることができるかどうかを示します。

コントロール・センターを使用して表を作成するには、以下のようにします。

1. オブジェクト・ツリーを順に展開し、**「表 (Tables)」**フォルダーを表示します。
2. **「表 (Tables)」**フォルダーを右クリックして、**「作成 (Create)」**をクリックします。
3. このウィザードのステップに従って、タスクをすべて実行します。

コマンド行を使用して表を作成するには、以下のように入力します。

```
CREATE TABLE <NAME>
   (<column_name>  <data_type>  <null_attribute>)
   IN <TABLE_SPACE_NAME)
```

RESOURCE 表スペースに EMPLOYEE 表を作成するための CREATE TABLE ステートメントの例を次に示します。この表はサンプル・データベースの中で定義されています。

```
CREATE TABLE EMPLOYEE
   (EMPNO     CHAR(6)     NOT NULL PRIMARY KEY,
    FIRSTNME  VARCHAR(12) NOT NULL,
    MIDINIT   CHAR(1)     NOT NULL WITH DEFAULT,
    LASTNAME  VARCHAR(15) NOT NULL,
    WORKDEPT  CHAR(3),
    PHONENO   CHAR(4),
    PHOTO     BLOB(10M)   NOT NULL)
IN RESOURCE
```

表を作成するときは、表の列を構造化タイプの属性に基づいたものとなるよう選ぶことができます。そのような表を「型付き表」といいます。

型付き表は、別の型付き表から列の一部を継承するように定義できます。そのような表を「副表」といい、副表から継承された表を「スーパーテーブル」といいま

す。型付き表とすべての副表を組み合わせたものを「表階層」といいます。表階層内の最上部にある表 (スーパーテーブルを持たない表) を階層の「ルート表」といいます。

グローバル一時表を宣言するには、DECLARE GLOBAL TEMPORARY TABLE ステートメントを使用します。

照会の結果に基づいて定義される表を作成することもできます。この種の表は、*マテリアライズ照会表* と呼ばれます。

関連概念:

- 「*データ移動ユーティリティー ガイドおよびリファレンス*」の『インポートの概要』
- 「*データ移動ユーティリティー ガイドおよびリファレンス*」の『ロードの概要』
- 「*データ移動ユーティリティー ガイドおよびリファレンス*」の『プラットフォーム間のデータの移動 - ファイル・フォーマットの考慮事項』
- 130 ページの『ユーザー定義タイプ (UDT)』

関連タスク:

- 136 ページの『マテリアライズ照会表の作成』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『CREATE TABLE ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『INSERT ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『DECLARE GLOBAL TEMPORARY TABLE ステートメント』
- 「コマンド・リファレンス」の『IMPORT コマンド』
- 「コマンド・リファレンス」の『LOAD コマンド』

表の作成とデータの読み込みに関する詳細

表には、データがすべて入っています。表を作成し、その中にデータを入れる際に考慮すべきことがたくさんあります。

表のスペース圧縮の紹介

ディスクへの保管時に、表が占めるスペースをできるだけ少なくするための方法は以下の 2 つです。

- 列値が NULL の場合、定義済みで固定された量のスペースをとっておかないようにします。
- 列値が簡単に分かるか、または決定できる場合 (デフォルト値のように)、また値がレコードのフォーマットおよび列の抽出中にデータベース・マネージャーに対して使用可能な場合。

DB2® Universal Database (DB2 UDB) には、このタイプのスペース節約を可能にする、オプションのレコード・フォーマットがあります。スペースの節約は、列レベルだけでなく、表レベルでも行えます。

関連概念:

- 「*管理ガイド: プランニング*」の『データベース・オブジェクトのスペース所要量』
- 100 ページの『新しい表のスペース圧縮』
- 183 ページの『既存の表のスペース圧縮』

新しい表のスペース圧縮

テーブルの作成時、オプションの VALUE COMPRESSION 文節を使用して、表が表レベル、そしておそらく列レベルで行フォーマットを節約するスペースを使用していることを指定します。

VALUE COMPRESSION が使用されると、NULL および定義済み可変長データ・タイプ (VARCHAR、 VARGRAPHICS、 LONG VARCHAR、 LONG VARGRAPHIC、 BLOB、 CLOB、および DBCLOB) に割り当てられた、長さがゼロのデータは、ディスク上に保管されません。これらのデータ・タイプに関連したオーバーヘッド値のみがディスク・スペースを使用します。

VALUE COMPRESSION が使用されると、COMPRESS SYSTEM DEFAULT オプションを使用して、さらにディスク・スペース使用量を削減することができます。挿入または更新値が、列のデータ・タイプのシステム・デフォルト値に等しい場合、使用されるディスク・スペースは最小になります。デフォルト値はディスク上に保管されません。 COMPRESS SYSTEM DEFAULT をサポートするデータ・タイプには、すべての数値タイプ列、固定長文字、および固定長 GRAPHIC ストリング・データ・タイプが含まれます。これは、ゼロおよびブランクは圧縮できるということです。

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『CREATE TABLE ステートメント』

ラージ・オブジェクト (LOB) 列に関する考慮事項

ラージ・オブジェクト列が入っている表を作成する前に、次の事柄を決定する必要があります。

1. LOB 列に対する変更をログに記録するか。

   これらの変更をログに記録したくない場合は、表の作成時に NOT LOGGED 文節を指定して、ロギングをオフにする必要があります。

   ```
   CREATE TABLE EMPLOYEE
      (EMPNO     CHAR(6)     NOT NULL PRIMARY KEY,
       FIRSTNME   VARCHAR(12)  NOT NULL,
       MIDINIT   CHAR(1)     NOT NULL WITH DEFAULT,
       LASTNAME   VARCHAR(15)  NOT NULL,
       WORKDEPT   CHAR(3),
       PHONENO    CHAR(4),
       PHOTO     BLOB(10M)   NOT NULL  NOT LOGGED)
   IN RESOURCE
   ```

   LOB 列の大きさが 1 GB を超える場合は、ロギングをオフにしなければなりません。 (目安として、大きさが 10 MB を超える LOB 列はログに記録すること

ができません。) 列定義に指定した他のオプションと同様、ロギング・オプションを変更するための唯一の方法は、表を再作成することです。

変更をログに記録することを選択しなくても、 LOB 列にはシャドー が作成されて、ロールバックがシステム生成エラーの結果であるか、アプリケーションの要求であるかによって、変更をロールバックすることができます。シャドーの作成は、現行のストレージの内容が上書きされない場所では、リカバリー技法となります。つまり、古くなった未修正ページは「シャドー」コピーとして保持されます。これらのコピーは、トランザクション・ロールバックのサポートに必要でなくなると、破棄されます。

注: RESTORE および ROLLFORWARD コマンドを使用してデータベースをリカバリーするとき、『NOT LOGGED』 になっており、最後のバックアップで書き込まれた LOB データは*バイナリー・ゼロで置き換えられます。*

2. LOB 列のために必要なスペースを最小化するか。

   CREATE TABLE ステートメントの COMPACT 文節を使用すると、 LOB 列をできる限り小さくすることができます。たとえば、以下のとおりです。

```
CREATE TABLE EMPLOYEE
   (EMPNO     CHAR(6)     NOT NULL PRIMARY KEY,
    FIRSTNME  VARCHAR(12) NOT NULL,
    MIDINIT   CHAR(1)     NOT NULL WITH DEFAULT,
    LASTNAME  VARCHAR(15) NOT NULL,
    WORKDEPT  CHAR(3),
    PHONENO   CHAR(4),
    PHOTO     BLOB(10M)   NOT NULL  NOT LOGGED  COMPACT)
IN RESOURCE
```

   特に LOB 値のサイズが (行わなければならないストレージ調整のために) 増えた場合には、 LOB 列を小さくした表を付加するときに、*パフォーマンス上のコスト* がかかります。

   予備ファイルの割り振りがサポートされておらず、 LOB が SMS 表スペースに置かれているプラットフォームでは、 COMPACT 文節を使用することを検討してください。予備ファイルの割り振りは、オペレーティング・システムが物理ディスク・スペースを使用する方法と関係があります。予備ファイルの割り振りをサポートするオペレーティング・システムは、予備ファイルの割り振りをサポートしないオペレーティング・システムに比べ、 LOB の保管にそれほどの物理ディスク・スペースを使用しません。 COMPACT オプションを指定すると、予備ファイルの割り振りがサポートされているかどうかに関係なく、大量の物理ディスク・スペースを「節約」することができます。 COMPACT を使用した場合には物理ディスク・スペースが節約できるので、オペレーティング・システムが予備ファイルの割り振りをサポートしていない場合には、 COMPACT を使用することを検討すべきです。

   注: DB2® システム・カタログは LOB 列を使用しており、以前のバージョンよりも多くのスペースを使用します。

3. DB2 システム・カタログ内の LOB 列も含む LOB 列について、パフォーマンスの向上を求めるか。

   カタログ表の中にラージ・オブジェクト (LOB) 列があります。 LOB データは他のデータと一緒にバッファー・プールの中には保持されず、必要になるたびに

ディスクから読み取られます。ディスクからの読み取りによって、カタログの LOB 列が含まれている場合には、 DB2 のパフォーマンスが低下します。ファイル・システムは通常データを保管 (またはキャッシュ) するための独自の場所を持っているので、 SMS 表スペースを使用するか、ファイル・コンテナー上に作成された DMS 表スペースを使用して、 LOB が以前に参照されている場合に入出力が発生する可能性を避けるようにしてください。

関連概念:

- 「*管理ガイド: プランニング*」の『ラージ・オブジェクト・データのスペース所要量』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『CREATE TABLE ステートメント』
- 「*SQL リファレンス 第 1 巻*」の『ラージ・オブジェクト (LOB)』

制約の定義

この節では、制約の定義方法について説明します。

- 『ユニーク制約の定義』
- 103 ページの『参照制約の定義』
- 107 ページの『表チェック制約の定義』
- 108 ページの『情報制約の定義』

制約の詳細については、「*管理ガイド: プランニング*」の制約の計画に関する節と、 *SQL リファレンス* を参照してください。

ユニーク制約の定義

ユニーク制約 は、指定されたキー内のそれぞれの値がユニークなものになるようにします。 1 つの表は、最大 1 つのユニーク制約を主キーとして定義して、任意の数のユニーク制約を持つことができます。

制約事項:

副表にユニーク制約を定義することはできません。

1 つの表当たり 1 つの主キーだけが可能です。

手順:

CREATE TABLE または ALTER TABLE ステートメントの UNIQUE 文節を使用してユニーク制約を定義します。ユニーク・キーは複数の列で構成できます。 1 つの表上で、複数のユニーク制約が許されます。

いったん確立されると、INSERT または UPDATE ステートメントが表内のデータを修正するときに、データベース・マネージャーによって、そのユニーク制約が自動的に施行されます。ユニーク制約は、ユニーク索引を通して施行されます。

ユニーク制約が ALTER TABLE ステートメントに定義され、そのユニーク・キーの同じ列のセット上に索引が存在する場合、その索引はユニーク索引となり、制約によって使用されます。

任意のユニーク制約を 1 つとって、それを主キー として使用することができます。主キーは、(他のユニーク制約と一緒に) 参照制約の中の親キーとして使用することができます。主キーを定義するには、 CREATE TABLE または ALTER TABLE ステートメントで PRIMARY KEY 文節を使います。主キーには、複数の列を含めることができます。

1 次索引は、主キーの値がユニークなものとなるように施行します。主キーの指定された表を作成すると、データベース・マネージャーはその主キーについての 1 次索引を作成します。

ユニーク制約として使用される索引に対するパフォーマンス上のヒントには、以下のものがあります。

- 索引のある空の表の初期ロードを実行するときは、 IMPORT よりも LOAD のほうがパフォーマンスはよくなります。これは、LOAD の INSERT または REPLACE のいずれのモードを使用していても同じです。
- 索引のある既存の表に、大量のデータを追加するときには (IMPORT INSERT または LOAD INSERT を使用)、 IMPORT よりも LOAD のほうがパフォーマンスは多少よくなります。
- IMPORT コマンドを使用して最初の大量のデータのロードを行う場合は、データがインポートされた後でユニーク・キーを作成してください。こうすれば、表のロード時に索引保守のためのオーバーヘッドを避けることができます。さらに、索引が使用する記憶域を最小限にすることができます。
- REPLACE モードでロード・ユーティリティーを使用している場合は、データをロードする前にユニーク・キーを作成してください。この場合、ロード中に索引を作成するほうが、ロードの後で CREATE INDEX ステートメントを使用するより効率的です。

関連概念:

- 「*SQL リファレンス 第 1 巻*」の『キー』
- 「*SQL リファレンス 第 1 巻*」の『制約』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『ALTER TABLE ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『CREATE TABLE ステートメント』

参照制約の定義

表定義と列定義に参照制約を追加すると、参照保全が課せられます。参照制約がデータベース・マネージャーに対して定義されると、表および列内のデータへの変更は、定義済み制約に対してチェックされます。要求アクションの完了は、制約検査の結果に依存します。

手順:

参照制約は、 CREATE TABLE または ALTER TABLE ステートメントの FOREIGN KEY 文節および REFERENCES 文節を使用して確立されます。型付き表に対する参照制約、または型付き表である親表に対する参照制約からの影響があります。

外部キーの指定によって、1 つの表の行内または 2 つの表の行間の値について、制約が施行されます。データベース・マネージャーは、表定義で指定された制約を検査し、それに応じて関係を維持します。その目標は、1 つのデータベース・オブジェクトが別のオブジェクトを参照するときに、いつでも整合性が保たれているようにすることです。

たとえば、主キーと外部キーは、それぞれ部署番号の列を持ちます。 EMPLOYEE 表の場合、列名は WORKDEPT であり、DEPARTMENT 表の場合、列名は DEPTNO です。この 2 つの表の間の関係は、次の制約によって定義されています。

- EMPLOYEE 表の各従業員の部署番号は 1 つだけであり、その番号は DEPARTMENT 表の中にも存在しています。
- EMPLOYEE 表の各行は、DEPARTMENT 表の 1 つの行だけと関連があります。表同士の間には、一意の関連があります。
- EMPLOYEE 表の行のうち WORKDEPT の値が NULL 値でないものは、それぞれ DEPARTMENT 表の DEPTNO 列の 1 つの行と関連しています。
- DEPARTMENT 表は親表であり、EMPLOYEE 表は従属表です。

親表である DEPARTMENT を定義する SQL ステートメントは、次のとおりです。

```
CREATE TABLE DEPARTMENT
   (DEPTNO    CHAR(3)     NOT NULL,
    DEPTNAME  VARCHAR(29) NOT NULL,
    MGRNO     CHAR(6),
    ADMRDEPT  CHAR(3)     NOT NULL,
    LOCATION  CHAR(16),
       PRIMARY KEY (DEPTNO))
IN RESOURCE
```

従属表である EMPLOYEE を定義する SQL ステートメントは、次のとおりです。

```
CREATE TABLE EMPLOYEE
   (EMPNO     CHAR(6)     NOT NULL PRIMARY KEY,
    FIRSTNME   VARCHAR(12)  NOT NULL,
    LASTNAME   VARCHAR(15)  NOT NULL,
    WORKDEPT   CHAR(3),
    PHONENO    CHAR(4),
    PHOTO     BLOB(10m)   NOT NULL,
       FOREIGN KEY DEPT (WORKDEPT)
       REFERENCES DEPARTMENT ON DELETE NO ACTION)
IN RESOURCE
```

DEPTNO 列を DEPARTMENT 表の主キーとして指定し、 WORKDEPT を EMPLOYEE 表の外部キーとして指定すると、 WORKDEPT 値についての参照制約を定義することになります。この制約は、2 つの表の間での値の参照保全を施行するものとなります。この場合、EMPLOYEE 表に追加される従業員の部署番号は、DEPARTMENT 表の中にあるものでなければなりません。

EMPLOYEE 表の参照制約の削除規則は、NO ACTION です。つまり、DEPARTMENT 表から部署を削除しようとしても、その部署に従業員がいれば削除はできません。

この例では CREATE TABLE ステートメントを使って参照制約を追加していますが、 ALTER TABLE ステートメントを使うこともできます。

別の例: 前の例の中で使用されたのと同じ表定義が使用されます。また、DEPARTMENT 表は、EMPLOYEE 表より前に作成されます。各部署にはマネージャーがおり、そのマネージャーは EMPLOYEE 表にリストされています。DEPARTMENT 表の MGRNO 番号は、実際には EMPLOYEE 表の外部キーになっています。この参照サイクルのために、この制約にはわずかに問題があります。外部キーは後で追加することができます。また、CREATE SCHEMA ステートメントを使用して、EMPLOYEE 表と DEPARTMENT 表の両方を同時に作成することもできます。

関連概念:

- 105 ページの『FOREIGN KEY 文節』
- 106 ページの『REFERENCES 文節』

関連タスク:

- 192 ページの『外部キーの追加』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『ALTER TABLE ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『CREATE SCHEMA ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『CREATE TABLE ステートメント』

FOREIGN KEY 文節

外部キーは、同じ表または別の表内の主キーまたはユニーク・キーを参照します。外部キーを割り当てると、指定した参照制約にしたがって参照保全が維持されます。外部キーを定義するには、 CREATE TABLE または ALTER TABLE ステートメントで FOREIGN KEY 文節を使います。

外部キーの中の列の数は、親表の対応する基本制約またはユニーク制約 (親キーと呼ばれる) の中の列の数と同じでなければなりません。さらに、キー列定義の対応する部分は、それぞれ同じデータ・タイプ、同じ長さでなければなりません。外部キーには、*制約名* を割り当てることができます。名前は、割り当てなくても自動的に割り当てられます。使いやすさのためには、自分で*制約名* を割り当て、システムが生成した名前は使用しないようにすることをお勧めします。

複合外部キーの値は、外部キーの各列の値が親キーの対応する列の値と**等しければ**、親キーの値と一致します。 NULL 値が含まれる外部キーは、親キーが定義上 NULL 値を持つことができないため、親キーの値と一致することはありません。しかし、外部キーの NULL 値は、NULL 値ではないどの部分の値とも無関係に常に有効です。

外部キー定義に適用される規則は、次のとおりです。

- 1 つの表に複数の外部キーが可能です。
- 外部キーのいずれかの部分が NULL 可能なら、その外部キーは NULL 可能です。
- 外部キーのいずれかの部分が NULL ならば、その外部キーの値は NULL です。

関連タスク:

- 102 ページの『ユニーク制約の定義』

- 103 ページの『参照制約の定義』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『ALTER TABLE ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『CREATE TABLE ステートメント』

REFERENCES 文節

REFERENCES 文節は、関係の中の親表を指定し、必要な制約を定義するためのものです。この文節は列定義の中に含めることもできますし、 CREATE TABLE または ALTER TABLE ステートメントの中に FOREIGN KEY 文節を伴う別個の文節として含めることもできます。

REFERENCES 文節を列制約として指定する場合、暗黙の列リストは、指定する列名で構成されることになります。複数の列にそれぞれ別個の REFERENCES 文節を使うことも可能ですし、 1 つの列で複数の文節を使うことも可能です。

REFERENCES 文節には削除規則が含まれています。この例の中で使用される規則は、ON DELETE NO ACTION です。つまり、部署に従業員が配属されているなら、その部署は削除できません。削除規則としては、このほかに ON DELETE CASCADE、ON DELETE SET NULL、および ON DELETE RESTRICT があります。

関連概念:

- 105 ページの『FOREIGN KEY 文節』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『ALTER TABLE ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『CREATE TABLE ステートメント』

ユーティリティー操作に対する影響

ロード・ユーティリティーは自己参照と従属表の制約検査をオフにして、これらの表をチェック・ペンディング状態にします。 ロード・ユーティリティーが完了してから、オフになっていたすべての表の制約検査をオンにする必要があります。たとえば、DEPARTMENT 表と EMPLOYEE 表だけがチェック・ペンディング状態になっている場合には、以下のステートメントを実行することができます。

```
SET INTEGRITY FOR DEPARTMENT, EMPLOYEE IMMEDIATE CHECKED
```

インポート・ユーティリティーは、参照保全によって次のような影響を受けます。

- オブジェクト表にそれ自体以外の従属表がある場合、 REPLACE および REPLACE CREATE 関数は使用できません。

  これらの関数を使用する場合は、まずその表が親表となっている外部キーをすべてドロップしてください。インポートが終了したら、ALTER TABLE ステートメントで外部キーを再作成してください。
- 自己参照制約が入っている表へのインポートが成功するかどうかは、行をインポートする順番によります。

関連概念:

- 「*データ移動ユーティリティー ガイドおよびリファレンス*」の『インポートの概要』
- 「*データ移動ユーティリティー ガイドおよびリファレンス*」の『ロードの概要』
- 「*データ移動ユーティリティー ガイドおよびリファレンス*」の『保全性違反のチェック』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『SET INTEGRITY ステートメント』

表チェック制約の定義

表のチェック制約は、表チェック制約が定義されている表の行ごとに適用される検索条件を指定するものです。表チェック制約がデータベース・マネージャーに対して定義されると、表内のデータへの挿入または更新は、定義済み制約に対してチェックされます。要求アクションの完了は、制約検査の結果に依存します。

手順:

表に表チェック制約を作成するには、表の作成時または変更時にチェック制約定義を表に対応付けます。この制約は、 INSERT または UPDATE ステートメントで表の中のデータを変更する時に自動的に活動化されます。表のチェック制約は、DELETE または SELECT ステートメントに影響を及ぼしません。チェック制約は型付き表に関連付けることができます。

制約名は、同じ CREATE TABLE ステートメント内に指定されている他の制約と同じものにすることはできません。制約名を指定しない場合は、その制約の 18 文字のユニーク ID がシステムによって生成されます。

表チェック制約は、キーの固有性でカバーできないデータ保全規則、または参照保全制約を施行するために使用されます。場合によっては、ドメイン検査を実施するために、表チェック制約を使用することもできます。 CREATE TABLE ステートメントで発行された次の制約は、すべての活動の開始日付が同じ活動の終了日付より後になっていないことを確認します。

```
CREATE TABLE EMP_ACT
   (EMPNO       CHAR(6)      NOT NULL,
    PROJNO     CHAR(6)      NOT NULL,
    ACTNO      SMALLINT     NOT NULL,
    EMPTIME    DECIMAL(5,2),
    EMSTDATE   DATE,
    EMENDATE   DATE,
    CONSTRAINT ACTDATES CHECK(EMSTDATE <= EMENDATE) )
IN RESOURCE
```

この例では CREATE TABLE ステートメントを使って表チェック制約を追加していますが、 ALTER TABLE ステートメントを使うこともできます。

関連概念:

- 「*SQL リファレンス 第 1 巻*」の『制約』

関連タスク:

- 193 ページの『表チェック制約の追加』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『CREATE TABLE ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『ALTER SERVER ステートメント』

情報制約の定義

情報制約 は、SQL コンパイラーが使用できる規則ですが、データベース・マネージャーによって強制することはできません。 SQL コンパイラーには、SQL ステートメントを、最適化でき、必要なデータへのアクセス・パスを改善する書式にトランスフォームする、照会再書き込みのステージが含まれます。制約の目的は、データベース・マネージャーによってデータの付加的な検査をすることではなく、照会パフォーマンスを向上させることです。

手順:

CREATE TABLE または ALTER TABLE ステートメントを使用して情報制約を定義します。これらのステートメント内で、参照保全またはチェック制約を追加します。その後、データベース・マネージャーによって制約が強制されるようにするかどうか、および制約を照会最適化に使用するかどうかを指定して、制約属性をそれらの制約に関連付けます。

関連概念:

- 「*SQL リファレンス 第 1 巻*」の『制約』
- 「*管理ガイド: パフォーマンス*」の『SQL コンパイラーの処理』
- 「*管理ガイド: パフォーマンス*」の『照会書き直しのメソッドとその例』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『ALTER TABLE ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『CREATE TABLE ステートメント』

新しい表に生成列を定義する

生成列は基本表に定義されます。生成列に格納される値は、式を使って計算された値です。挿入操作や更新操作で指定された値ではありません。

手順:

作成する表で、特定の式や述部を頻繁に使用することが分かっている場合、その表に 1 つまたは複数の生成列を追加することができます。生成列を使用すると、表データを照会する際のパフォーマンスを向上させることができます。

たとえば、パフォーマンスが重要な場合、式の評価の費用を高める恐れのある要因が 2 つあります。

1. 照会中に式の評価を何度も行わなければならないこと。
2. 計算が複雑であること。

照会のパフォーマンスを向上させるには、式の結果を入れるために、列をもう 1 つ定義することができます。そうすれば、同じ式を含んだ照会を発行するときに、生成列を直接に使用できます。あるいは、オプティマイザーの照会書き直しコンポーネントで、その式を生成列に置き換えることもできます。

生成列には、非ユニークな索引を作成することもできます。

照会時に複数の表のデータを結合する場合、生成列を追加すると、オプティマイザーがより良い結合の方針を選択できるかもしれません。

以下に、CREATE TABLE ステートメントを使って生成列を定義する方法の例を示します。

```
CREATE TABLE t1 (c1 INT,
                 c2 DOUBLE,
                 c3 DOUBLE GENERATED ALWAYS AS (c1 + c2)
                 c4 GENERATED ALWAYS AS
                    (CASE WHEN c1 > c2 THEN 1 ELSE NULL END))
```

この表を作成した後で、生成列を使用して索引を作成できます。たとえば、

```
CREATE INDEX i1 ON t1(c4)
```

生成列を照会に利用できます。たとえば、

```
SELECT COUNT(*) FROM t1 WHERE c1 > c2
```

これは、以下のように書き換えることができます。

```
SELECT COUNT(*) FROM t1 WHERE c4 IS NOT NULL
```

別の例を以下に示します。

```
SELECT c1 + c2 FROM t1 WHERE (c1 + c2) * c1 > 100
```

これは、以下のように書き換えることができます。

```
SELECT c3 FROM t1 WHERE c3 * c1 > 100
```

生成列は、照会のパフォーマンスを向上させるために使用します。したがって、おそらく、生成列を追加するのは、表を作成したり、表にデータを読み込んだ後になるでしょう。

関連タスク:

- 197 ページの『既存の表での生成列の定義』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『CREATE INDEX ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『CREATE TABLE ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『SELECT ステートメント』

ユーザー定義の一時表の作成

ユーザー定義の一時表は、データベース中のデータを処理するために書き込みを行っているアプリケーションに必要です。データの操作の結果は、一時的に表に保管する必要があります。

前提条件:

USER TEMPORARY 表スペースは、ユーザー定義の一時表が作成される前に、存在しなくてはなりません。

制約事項:

この表の記述は、システム・カタログには現れません。したがって、この表を他のアプリケーションのために保持したり、他のアプリケーションと共用したりすることはできません。

この表を使用するアプリケーションが終了したりデータベースから切断されたりすると、表の中のデータはすべて削除され、表は暗黙的にドロップされます。

ユーザー定義一時表は、以下のものをサポートしません。

- LOB タイプの列 (または LOB に基づく特殊タイプ列)
- ユーザー定義タイプの列
- LONG VARCHAR 列
- DATALINK 列

手順:

一時表を定義するには、DECLARE GLOBAL TEMPORARY TABLE ステートメントを使用します。このステートメントは 1 つのアプリケーションの中から使用されます。ユーザー定義一時表が保持されるのは、アプリケーションがデータベースから切断されるまでの間だけです。

一時表を定義する方法の例を以下に示します。

```
DECLARE GLOBAL TEMPORARY TABLE gbl_temp
   LIKE empltabl
   ON COMMIT DELETE ROWS
   NOT LOGGED
   IN usr_tbsp
```

このステートメントにより、gbl_temp というユーザー一時表が作成されます。このユーザー一時表の列は、名前と記述が empltabl の列と完全に一致するように定義されています。暗黙定義には、列名、データ・タイプ、NULL 可能特性、および列のデフォルト値の属性だけが含まれます。他のすべての列属性 (ユニーク制約、外部キー制約、トリガー、索引を含む) は、定義されていません。 COMMIT 操作を実行すると、表で WITH HOLD カーソルがオープンしていなければ、表の中のデータはすべて削除されます。ユーザー一時表に対する変更内容はログに記録されません。ユーザー一時表は、指定された USER TEMPORARY 表スペースに置かれます。この表スペースがないと、この表の宣言は失敗します。

ROLLBACK または ROLLBACK TO SAVEPOINT がこの表の作成時に指定されると、表中のすべての行を削除する (DELETE ROWS、これがデフォルトです) か、表の行を保存する (PRESERVE ROWS) ことを指定できます。

関連タスク:

- 89 ページの『USER TEMPORARY 表スペースの作成』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『ROLLBACK ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『SAVEPOINT ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『DECLARE GLOBAL TEMPORARY TABLE ステートメント』

新しい表での ID 列の定義

ID 列 を使用すると、表に追加される個々の行に対し、ユニークな数値を DB2 が自動的に生成します。表に追加する個々の行を固有に識別する必要があることが分かっている場合、その表を作成する際に、ID 列を追加できます。表に追加する個々の行のユニークな数値を保証するには、ID 列にユニーク索引を作成するか、または主キーとして宣言する必要があります。

制約事項:

いったん作成した後で、表の記述を変更して、ID 列を組み込むことはできません。

行が表に、指定された明示的な ID 列値で挿入される場合、次の内部生成される値は更新されず、表内の既存の値と競合する可能性があります。 ID 列の値のユニーク性が、その ID 列に定義されている主キーまたはユニーク索引によって強制される場合、重複値によってエラー・メッセージが生成されます。

手順:

CREATE TABLE ステートメントで AS IDENTITY 文節を使用すると、ID 列を指定できます。

以下に、CREATE TABLE ステートメントを使って ID 列を定義する方法の例を示します。

```
CREATE TABLE table (col1 INT,
                    col2 DOUBLE,
                    col3 INT NOT NULL GENERATED ALWAYS AS IDENTITY
                                      (START WITH 100, INCREMENT BY 5))
```

この例では、3 番目の列が ID 列です。この列で使用される値を指定して、追加される個々の行を固有に識別することもできます。この例の場合、最初に入力される行については、値 ″100″ が ID 列に入れられます。この表に行が追加されるごとに、値は 5 ずつ増えます。

ID 列を使用する他の例としては、注文番号、従業員番号、在庫番号、事例番号などがあります。 ALWAYS または BY DEFAULT を指定して、 DB2 によって ID 列の値が生成されるようにすることができます。

GENERATED ALWAYS として定義された ID 列は、常に DB2 が生成する値に指定されます。アプリケーションが明示的に値を指定することはできません。 ID 列を GENERATED BY DEFAULT として定義すると、アプリケーションが明示的に ID 列の値を指定できます。アプリケーションが値を指定しないと、DB2 が値を生成します。アプリケーションが値を制御するので、値が固有であることを DB2 が保証することはできません。 GENERATED BY DEFAULT 文節は、既存の表の内容をコピーする目的でデータ伝搬を行う場合や、表のアンロードや再ロードを行う場合に使用します。

関連概念:

- 113 ページの『ID 列とシーケンスの比較』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『CREATE TABLE ステートメント』

シーケンスの作成

シーケンス とは、値の自動生成を可能にするデータベース・オブジェクトです。シーケンスは、ユニーク・キー値を生成するタスクに最も適しています。アプリケーションはシーケンスを使用し、データベースの外部にユニーク・カウンターを生成したことによって発生する可能性のある、並行性およびパフォーマンスの問題を回避することができます。

制約事項:

ID 列属性とは異なり、シーケンスは特定の表列に関連付けられたり、ユニークな表列にバインドされることはなく、その表列からのみアクセス可能です。

シーケンスを含むデータベースを以前の状態にリカバリーすると、いくつかのシーケンスで値が重複する場合があります。値の重複を回避するため、シーケンスを含むデータベースを以前の状態にリカバリーしないでください。

NEXTVAL または PREVVAL 式が使用できるところでは、いくつかの制約事項があります。

手順:

以下の方法のいずれかでシーケンスが値を生成するよう、シーケンスを作成または変更することができます。

- バインドなしで単調増分または減分する
- ユーザー定義の制限まで単調増分または減分して終了する
- ユーザー定義の制限まで単調増分または減分し、先頭に戻って循環する

以下に、シーケンス・オブジェクト作成の例を示します。

```
CREATE SEQUENCE order_seq
   START WITH 1
   INCREMENT BY 1
   NOMAXVALUE
   NOCYCLE
   CACHE 24
```

この例で、シーケンスは order_seq です。 1 から始まり、上限なしで 1 ずつ増えていきます。上限が割り当てられていないため、先頭に戻って循環することはありません。 CACHE パラメーターに関連する数値は、データベース・マネージャーが事前割り当てし、メモリーに保管するシーケンス値の最大数を指定します。

生成されるシーケンス番号のプロパティー

- 値は位取りがゼロの数値データ・タイプになります。このようなデータ・タイプは SMALLINT、BIGINT、INTEGER、および DECIMAL です。
- 連続値は、指定した整数増分値によって異なる場合があります。デフォルト増分値は 1 です。
- カウンター値はリカバリー可能です。カウンター値は、リカバリーが要求されたときにログから再構成されます。
- パフォーマンスを上げるため、値をキャッシュに入れることができます。値を事前割り振りしてキャッシュに保管しておくと、シーケンスのために値を生成する

とき、ログへの非同期入出力が少なくなります。システム障害イベントが発生した場合、コミットされていないキャッシュ値はすべて使用されなくなり、失われたものと見なされます。 CACHE に指定された値は、失われる可能性のあるシーケンス値の最大数です。

シーケンスで使用される式には、以下の 2 つがあります。

PREVVAL 式は、現行アプリケーション・プロセス内の直前のステートメントに指定されたシーケンスについて最後に生成された値を返します。

NEXTVAL 式は、指定されたシーケンスの次の値を返します。 NEXTVAL 式がシーケンスの名前を指定していれば、新しいシーケンス番号が生成されます。ただし、照会の中に同じシーケンス名を指定している NEXTVAL 式のインスタンスが複数ある場合、シーケンスのカウンターは結果の行ごとに 1 つずつ増えていき、NEXTVAL のすべてのインスタンスが結果の行に同じ値を戻します。

同じシーケンス番号は、先頭の行の NEXTVAL 式およびその他の行の PREVVAL 式を使用してシーケンス番号を参照することによって、 2 つの異なる表内のユニーク・キー値として使用することができます。

たとえば、以下のとおりです。

```
INSERT INTO order (orderno, custno)
   VALUES (NEXTVAL FOR order_seq, 123456);
INSERT INTO line_item (orderno, partno, quantity)
   VALUES (PREVVAL FOR order_seq, 987654, 1)
```

関連概念:

- 113 ページの『ID 列とシーケンスの比較』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『CREATE SEQUENCE ステートメント』

ID 列とシーケンスの比較

ID 列とシーケンスは類似していますが、異なる点もあります。それぞれの特性は、データベースおよびアプリケーションの設計時に利用することができます。

ID 列の特性

- ID 列は、表の作成時にのみ、表の一部として定義することができます。表を作成した後、変更して ID 列を追加することはできません。 (ただし、既存の ID 列の特性を変更することはできます。)
- ID 列は、1 つの表の値を自動的に生成します。
- ID 列が GENERATED ALWAYS として定義されている場合、使用される値は常にデータベース・マネージャーによって生成されます。表の内容の変更中にアプリケーションで独自の値を指定することはできません。

シーケンス・オブジェクトの特性

- シーケンス・オブジェクトとは、どの表にも関連付けられていないデータベース・オブジェクトです。

- シーケンス・オブジェクトは、SQL ステートメントで使用可能な順次値を生成します。
- シーケンス・オブジェクトはどのアプリケーションでも使用できるため、指定されたシーケンス内の次の値、およびステートメントの実行前に生成された値の検索を制御するための 2 つの式があります。 PREVVAL 式は、現行セッション内の直前のステートメントに指定されたシーケンスについて最後に生成された値を返します。 NEXTVAL 式は、指定されたシーケンスの次の値を返します。これらの式を使用すると、複数の表内の複数の SQL ステートメントで同じ値を使用できるようになります。

以上がこれら 2 つの特性のすべてではありませんが、このような特性を考慮することによって、データベース設計やデータベースで使用するアプリケーションに応じてどちらを使用すればよいか判断するために役立ちます。

関連タスク:

- 108 ページの『新しい表に生成列を定義する』
- 112 ページの『シーケンスの作成』
- 197 ページの『既存の表での生成列の定義』

範囲クラスター表の例

以下の 2 つの例は、範囲クラスター表の作成方法を示す簡単な例です。これらの例では、表のキーとして単一列または複数列を使用する方法を紹介します。加えて、データをオーバーフローできる表とデータをオーバーフローできない表の作成方法も示します。

1 つ目の例は、STUDENT_ID を使用して生徒を探し出すときに使用する範囲クラスター表を示しています。各生徒のレコードには、以下の情報が含まれます。

- 学校 ID
- プログラム ID
- 生徒番号
- 生徒 ID
- 生徒のファーストネーム
- 生徒のラストネーム
- 生徒の成績平均点 (GPA)

この場合は、生徒のレコードを扱うすべての処理を、STUDENT_ID に基づいて行おうとしています。 STUDENT_ID は、生徒のレコードの追加、更新、および削除に使用されます。

注: その他の索引は、別の時に別個に追加できます。ただし、この例の目的からして、表の編成と表のデータへのアクセス方法は、表の作成時に定義されます。

以下は、この表に必要な構文です。

```
CREATE TABLE STUDENTS
   (SCHOOL_ID      INT NOT NULL,
    PROGRAM_ID       INT NOT NULL,
    STUDENT_NUM    INT NOT NULL,
```

```
    STUDENT_ID     INT NOT NULL,
    FIRST_NAME     CHAR(30),
    LAST_NAME      CHAR(30),
    GPA            FLOAT)
   ORGANIZE BY KEY SEQUENCE
   (STUDENT_ID     STARTING FROM 1 ENDING AT 1000000)
    ALLOW OVERFLOW
;
```

各レコードのサイズは、列の合計です。この場合、10 バイトのヘッダー + 4 + 4 + 4 + 4 + 30 + 30 + 8 + 3 (ヌル可能列の場合) で、 97 バイトになります。ページ・サイズが 4 KB (つまり 4096 バイト) であれば、オーバーヘッドを計算した結果が 4038 バイトありますから、ページごとに 42 レコード分のスペースがあることになります。 100 万人の生徒のレコードを保管できるようにする場合は、 100 万をページ当たりのレコード数 42 で除算して、23809.5 ページが必要になります。つまり、切り上げて 23810 ページが必要、ということです。これに、表のオーバーヘッド分 4 ページと、エクステント・マッピング用の 3 ページが追加されます。結果として、4 KB のページを 23817 ページ事前割り振りすることが必要になります。 (エクステント・マッピングは、この表が 1 つのコンテナーに保持されることを前提としています。マッピング用のページは、コンテナーごとに 3 ページ必要です。)

2 つ目の例は、1 つ目の例を少し変えたもので、教育委員会を念頭に置いています。教育委員会に属する学校は 200 あり、そのそれぞれには、 35 人の生徒を定員とする 20 のクラスルームがあります。これは、この教育委員会が最大 140,000 人の生徒に対応できる計算になります。

この場合は、生徒のレコードを 3 つの要素、つまり SCHOOL_ID、CLASS_ID、および STUDENT_NUM の値に基づいて処理することにします。これらの 3 つの列のそれぞれは、ユニークな値を持ち、合わせて生徒のレコードの追加、更新、および削除に使用されます。

注: 1 つ目の例と同様、その他の索引は、別のときに別個に追加できます。

以下は、この表に必要な構文です。

```
CREATE TABLE STUDENTS
   (SCHOOL_ID      INT NOT NULL,
    CLASS_ID       INT NOT NULL,
    STUDENT_NUM    INT NOT NULL,
    STUDENT_ID     INT NOT NULL,
    FIRST_NAME     CHAR(30),
    LAST_NAME      CHAR(30),
    GPA            FLOAT)
   ORGANIZE BY KEY SEQUENCE
   (SCHOOL_ID      STARTING FROM 1 ENDING AT 200,
    CLASS_ID       STARTING FROM 1 ENDING AT 20,
    STUDENT_NUM    STARTING FROM 1 ENDING AT 35)
    DISALLOW OVERFLOW
;
```

この例では、オーバーフローは許可されません。これは、教育委員会で、方針として、各クラスの生徒数に制限を設けることが少なくないゆえに意味のある例です。この例の場合、クラスのサイズは最大でも 35 より大きくはなり得ません。この要

素と、クラスルームや学校の数によって発生する物理的な限界を合わせて考えると、教育委員会に属する生徒の数にオーバーフローがありえないことは明らかです。

学校のクラスルームの数が変動することは、可能性としてありえることです。もしそのような場合には、クラスルーム数の範囲を定義する (CLASS_ID を使用) 際に、すべての学校を考慮して、クラスルームの最大数を上限に設定する必要があります。これは、比較的小さい学校 (大きな学校よりもクラスルームの数が少ない) にとっては、使用されることのない生徒レコード用のスペースが発生する可能性があることを意味します (ただし、たとえば、仮設のクラスルームが学校に追加される場合を除く)。

1 つ目の例と同じ 4 KB のページ・サイズと同じ生徒レコードのサイズで計算すると、 1 ページあたりのレコード数として 42 が算出できます。生徒のレコードが 140,000 ですから、3333.3 ページ、切り上げて 3334 ページが必要になります。なお、表情報用のページが 2 ページ、エクステント・マッピング用のページが 3 ページ必要です。結果として、4 KB のページを 3339 ページ事前割り振りすることが必要になります。

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『CREATE TABLE ステートメント』

SQL コンパイラーによる範囲クラスター表の処理方法

SQL コンパイラーは、 2 次 B+ ツリー索引を持つ通常の表と同様の方法で範囲クラスター表 (RCT) を処理します。 B+ ツリー索引を通してレコードのロケーションやレコード ID (RID) を決定するのではなく、 RCT は、レコード・キー値に関係した機能的な検索と範囲定義のアルゴリズムを使用します。これは、RID をすばやく取得するためにキー値を使用するため、索引を持つのと似ています。

要求されたデータへの最良のアクセス・パスを決定しようと機能するとき、 SQL コンパイラーは、表に関して保持されている統計情報を使用します。 RUNSTATS コマンドが出されると、表のスキャンの間に、索引統計が収集されます。 RCT の場合、表は通常の表としてモデル化され、索引は機能ベースの索引としてモデル化されます。

範囲クラスター表の作成でオーバーフローが許可されている場合は、表内でのレコードの順序は保証されません。

関連概念:

- 「*管理ガイド: プランニング*」の『範囲クラスター表』
- 116 ページの『範囲クラスター表使用のガイドライン』

範囲クラスター表使用のガイドライン

DB2® Universal Database および範囲クラスター表 (RCT) を扱う際は、以下のガイドラインに注意してください。

- キー値の範囲を定義する際、最小値の指定はオプションです。指定がない場合、最小値はデフォルトで 1 になります。最小値や最大値に負の値を指定することも

可能です。負の値を扱う場合は、必ず最小値を明示的に宣言してください。たとえば、ORGANIZE BY KEY SEQUENCE (F1 STARTING FROM -100 ENDING AT -10) のようにします。

- 範囲クラスター表の定義に使用されたのと同じキー値で通常の索引を作成することはできません。
- 範囲クラスター表では、一部の ALTER TABLE オプションが使用できません。表の物理構造に影響を与えないオプションは使用できます。
- 範囲クラスター表の作成プロセスでは必要なディスク・スペースの事前割り振りが行われるため、その分のスペースが使用可能でないと、表の作成は失敗します。

関連概念:

- 「*管理ガイド: プランニング*」の『範囲クラスター表』
- 114 ページの『範囲クラスター表の例』

表のディメンションの定義

ディメンション は、表のクラスタリング・キーです。 1 つ以上のディメンションを 1 つの表に選択できます。表に複数のディメンションがある場合、その表は複数ディメンションでクラスター化された表と見なされます。そのような表は、ORGANIZE BY DIMENSIONS 文節を指定した CREATE TABLE ステートメントを使って作成されます。

制約事項:

ORGANIZE BY [DIMENSIONS] 文節で使用される列セットは、 CREATE INDEX ステートメントの規則に従う必要があります。列は、ストレージ中のデータの物理順序を保守するのに使用されるキーとして扱われます。

手順:

各ディメンションは、ORGANIZE BY [DIMENSIONS] 文節および 1 つ以上の列を使用して、 CREATE TABLE ステートメントで指定されます。括弧は、ディメンション・リスト内で単一のディメンションに関連したグループ列をリストするのに使用されます。

データは、1 つ以上のディメンションで物理的にクラスター化されており、同時にそれと同じ数のディメンションが指定されます。データは、ディメンション行に沿ってエクステントまたは「ブロック」により編成されます。ディメンションの述部を使用してデータを照会すると、スキャンは、関係するディメンション値を含む表のエクステントだけに限定されることがあります。さらに、エクステントはディスク上の順次ページのセットなので、これらのスキャンに非常に効率的なプリフェッチが実行できることもあります。

単一のクラスタリング索引を持つ表は、時間の経過とともに表スペースがいっぱいになり、非クラスタリングされる可能性がありますが、複数のディメンションをもつ表は、すべてのディメンションにわたって自動的かつ連続的にクラスタリングを保守することができます。その結果、データに対して順序をリストアするために表を再編成する必要がありません。

ディメンション・ブロック索引は、指定されるディメンションごとに自動的に作成されます。ディメンション・ブロック索引は、ディメンションに沿ってデータにアクセスするために使用されます。これは個々の行ではなくエクステントへのポインターになるので、正規の索引よりはかなり小さくなります。これらのディメンション・ブロック索引は、特定のディメンション値を含む表のエクステントのみに、すばやくアクセスするために使用できます。

複合ブロック索引は、すべてのディメンション・キー列を含んでおり、自動的に作成されます。複合ブロック索引は、挿入および更新活動中にデータのクラスタリングを保守するのに使用されます。複合ブロック索引は、照会処理において、表内の特定の複数のディメンション値を持つデータにアクセスするためにも使われます。

注: 複合ブロック索引のキー部分の順序は、その使用法、または照会処理の適用度に影響する場合があります。キー部分の順序は、MDC 表の作成時に使われる ORGANIZE BY [DIMENSIONS] 文節全体の中にある列の順序によって決定されます。たとえば、以下の方法で表が作成される場合を考えてみます。

```
CREATE TABLE t1 (c1 int, c2 int, c3 int, c4 int)
   ORGANIZE BY DIMENSIONS (c1, c4, (c3,c1), c2)
```

この場合、複合ブロック索引は列 (c1,c4,c3,c2) で作成されます。 c1 はディメンション文節で 2 度指定されていますが、複合ブロック索引のキー部分では 1 度しか使用されず、最初に検出された順序で使用されます。挿入処理の場合、複合ブロック索引内のキー部分の順序は無関係ですが、照会処理の場合には順序が影響を与える可能性があります。したがって、複合ブロック索引の列の順序を (c1,c2,c3,c4) にした方が望ましければ、表は次のように作成するべきです。

```
CREATE TABLE t1 (c1 int, c2 int, c3 int, c4 int)
   ORGANIZE BY DIMENSIONS (c1, c2, (c3,c1), c4)
```

複合ブロック索引は、複合ブロック索引が持つすべての列が指定済みのディメンションにすでに入っている場合、作成されません。たとえば、複合ブロック索引は、次の表には作成されません。

```
CREATE TABLE t1 (c1 int, c2 int)
   ORGANIZE BY DIMENSIONS (c1,(c2,c1))
```

関連概念:

- 「*管理ガイド: プランニング*」の『マルチディメンション・クラスター化索引』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『ALTER TABLE ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『CREATE TABLE ステートメント』

階層表または型付き表の作成

階層表とは、型付き表階層の実装に関連付けられた表であり、階層のルート表と同時に作成されます。

構造化タイプ階層の作成の一環として、型付き表を作成することでしょう。型付き表は、特性が CREATE TYPE ステートメントで定義されるオブジェクトのインスタンスを保管するために使用されます。

前提条件:

階層表や型付き表が作成される型は、あらかじめ存在していなければなりません。

手順:

CREATE TABLE ステートメントの変種を使用して、階層表または型付き表を作成できます。

関連概念:

- 「*アプリケーション開発ガイド サーバー・アプリケーションのプログラミング*」の『型付き表』

関連タスク:

- 119 ページの『型付き表へのデータの読み込み』
- 136 ページの『型付きビューの作成』
- 「*アプリケーション開発ガイド サーバー・アプリケーションのプログラミング*」の『構造化型階層の作成』
- 「*アプリケーション開発ガイド サーバー・アプリケーションのプログラミング*」の『型付き表のドロップ』
- 「*アプリケーション開発ガイド サーバー・アプリケーションのプログラミング*」の『型付き表の作成』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『CREATE TABLE ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『CREATE TYPE (構造化) ステートメント』

型付き表へのデータの読み込み

構造型階層の確立の一環として、型付き表を作成することでしょう。型付き表は、特性が CREATE TYPE ステートメントで定義されるオブジェクトのインスタンスを保管するために使用されます。作成後、データをその型付き表に置く必要があります。

前提条件:

型付き表が存在している必要があります。

手順:

構造型を作成して、対応する表および副表を作成したら、型付き表にデータを読み込むことができます。

関連概念:

- 「*アプリケーション開発ガイド サーバー・アプリケーションのプログラミング*」の『型付き表での代理性』
- 「*アプリケーション開発ガイド サーバー・アプリケーションのプログラミング*」の『型付き表』

関連タスク:

- 118 ページの『階層表または型付き表の作成』
- 「*アプリケーション開発ガイド サーバー・アプリケーションのプログラミング*」の『型付き行へのオブジェクトの保管』
- 「*アプリケーション開発ガイド サーバー・アプリケーションのプログラミング*」の『型付き表のドロップ』
- 「*アプリケーション開発ガイド サーバー・アプリケーションのプログラミング*」の『型付き表の作成』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『CREATE TYPE (構造化) ステートメント』

複数の表スペースでの表の作成

表データは、表の索引や、表に関連した長形式列データと同じ表スペースに格納することができます。また索引や長形式の列データを、それ以外の表データとは別の表スペースに入れることもできます。

前提条件:

CREATE TABLE ステートメントを実行する前に、すべての表スペースが存在していなければなりません。

制約事項:

表の一部を分離することができるのは、DMS 表スペースを使用している場合のみです。

手順:

コントロール・センターを使用して複数の表スペースに 1 つの表を作成するには、以下のようにします。

1. オブジェクト・ツリーを順に展開し、**「表 (Tables)」**フォルダーを表示します。
2. **「表 (Tables)」**フォルダーを右クリックして、ポップアップ・メニューから**「作成 (Create)」**を選択します。
3. 表名を入力して、**「次へ (Next)」**をクリックします。
4. 表の列を選択します。
5. **「表スペース (Table space)」**ページで、**「別の索引スペースを使用 (Use separate index space)」**と**「ロング・スペースの分離の使用 (Use separate long space)」**をクリックし、情報を指定して、**「完了 (Finish)」**をクリックします。

コマンド行を使用して複数の表スペースに 1 つの表を作成するには、以下のように入力します。

```
CREATE TABLE <name>
   (<column_name>  <data_type>  <null_attribute>)
    IN <table_space_name>
    INDEX IN <index_space_name>
    LONG  IN <long_space_name>
```

次の例では、別の表スペースに表の別の部分を格納するために、 EMP_PHOTO 表が作成される方法を示します。

```
CREATE TABLE EMP_PHOTO
   (EMPNO        CHAR(6)      NOT NULL,
    PHOTO_FORMAT VARCHAR(10)  NOT NULL,
    PICTURE      BLOB(100K) )
IN RESOURCE
INDEX IN RESOURCE_INDEXES
LONG  IN RESOURCE_PHOTO
```

この例では、EMP_PHOTO データが次のように作成されます。

- EMP_PHOTO 表について作成された索引が RESOURCES_INDEXES 表スペースに格納される。
- PICTURE 列のデータが RESOURCE_PHOTO 表スペースに格納される。
- EMPNO および PHOTO_FORMAT 列のデータが RESOURCE 表スペースに格納される。

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『CREATE TABLE ステートメント』

パーティション・データベースでの表の作成

パーティション・データベースで幾つかの区画にまたがった表を作成することには、パフォーマンス上の利点があります。データの検索に関連した作業は、データベース・パーティションの中で分割することができます。

前提条件:

物理的に分割つまりパーティション化される表を作成する前に、以下のことを考慮する必要があります。

- 表スペースは、複数のデータベース・パーティションにわたって展開することができます。展開する区画の数は、データベース・パーティション・グループの中の区画の数によって決まります。
- 表は、同じ表スペースに置かれるか、または、最初の表スペースに加えて、同じデータベース・パーティション・グループに関連する別の表スペースの中に置かれることによって、併置を行うことができます。

制約事項:

*後から変更*することはできないため、注意深く適切なパーティション・キーを選択する必要があります。さらに、何らかのユニーク索引を (したがって、ユニーク・キーまたは主キーも)、パーティション・キーのスーパーセットとして定義しなければなりません。つまり、パーティション・キーが定義された場合、ユニーク・キーおよび主キーは、パーティション・キーと同じ列をすべて含まなければなりません (ユニーク・キーと主キーには、それ以上の列が含まれる場合があります)。

1 つの表の 1 区画についてのサイズの限界は、 64 GB または使用可能ディスク・スペースのうち、いずれか小さいほうになります。 (表スペースに 4 KB のページ・サイズを想定しています。) 表のサイズは、データベース・パーティションの数の 64 GB (または使用可能ディスク・スペース) 倍までの大きさにすることができます。表スペースのページ・サイズが 8 KB の場合、表のサイズは、データベー

ス・パーティションの数の 128 GB (または使用可能ディスク・スペース) 倍までの大きさにすることができます。表スペースのページ・サイズが 16 KB の場合、表のサイズは、データベース・パーティションの数の 256 GB (または使用可能ディスク・スペース) 倍までの大きさにすることができます。表スペースのページ・サイズが 32 KB の場合、表のサイズは、データベース・パーティションの数の 512 GB (または使用可能ディスク・スペース) 倍までの大きさにすることができます。

手順:

表を作成している時、いくつかのデータベース・パーティションの一部になる表を作成するように指定されます。パーティション・データベース環境で表を作成する場合、*パーティション・キー* という、追加のオプションがあります。パーティション・キーは、表の定義の一部であるキーです。このキーは、各データ行が保管される区画を判別します。

パーティション・キーを明示して指定しない場合、以下のデフォルトが使用されます。*デフォルトのパーティション・キーが適切であることを確認してください。*

- 主キーが CREATE TABLE ステートメントに指定された場合、主キーの最初の列がパーティション・キーとして使用されます。
- 主キーがない場合、ロング・フィールドでない最初の列が使用されます。
- デフォルトのパーティション・キーの要件を満たす列がない場合、表はパーティション・キーなしで作成されます (これは、単一パーティションのデータベース・パーティション・グループでのみ許されます)。

以下はその例です。

```
CREATE TABLE MIXREC (MIX_CNTL INTEGER NOT NULL,
                     MIX_DESC CHAR(20) NOT NULL,
                     MIX_CHR  CHAR(9) NOT NULL,
                     MIX_INT INTEGER NOT NULL,
                     MIX_INTS SMALLINT NOT NULL,
                     MIX_DEC DECIMAL NOT NULL,
                     MIX_FLT FLOAT NOT NULL,
                     MIX_DATE DATE NOT NULL,
                     MIX_TIME TIME NOT NULL,
                     MIX_TMSTMP TIMESTAMP NOT NULL)
                     IN MIXTS12
                     PARTITIONING KEY (MIX_INT) USING HASHING
```

上記の例で、表スペースは MIXTS12 であり、パーティション・キーは MIX_INT です。パーティション・キーが明示して指定されない場合、パーティション・キーは MIX_CNTL になります。 (主キーが指定されず、パーティション・キーが定義されない場合、パーティション・キーは、リスト内の最初のロング列以外の列になります。)

1 つの表の 1 つの行、およびその行に関するすべての情報は、常に同じデータベース・パーティション上に常駐します。

関連概念:

- 「*管理ガイド: プランニング*」の『データベース・パーティション・グループ』
- 「*管理ガイド: プランニング*」の『データベース・パーティション・グループの設計』
- 「*管理ガイド: プランニング*」の『表の併置』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『CREATE TABLE ステートメント』

トリガーの作成

トリガーは、指定した基本表または型付き表に対する INSERT、UPDATE、DELETE 文節と一緒に実行される一連のアクション、またはそれらの文節によってトリガー起動される一連のアクションを定義するものです。トリガーは、たとえば次のような目的で使います。

- 入力データの妥当性検査
- 新しく挿入された行の値を生成する
- 相互参照のために他の表から読み込む
- 監査履歴のために他の表に書き込む

トリガーを使えば、一般的な保全規則や業務規則をサポートできます。たとえば、トリガーによって、注文に応じる前に顧客のクレジット限度を調べたり、サマリー・データ表を更新したりできます。

トリガーを使うことの利点は、次のとおりです。

- アプリケーション開発がより速くなります。トリガーはデータベースの中に保管されるため、各アプリケーションの中にアクションをコーディングする必要がありません。
- 保守が簡単: 一度トリガーを定義すると、トリガーを作成した表にアクセスするたびに、そのトリガーが自動的に呼び出されます。
- 業務規則がグローバルに適用されます。業務方針が変わった場合、各アプリケーション・プログラムを変更しなくても、トリガーを変更するだけで済みます。

制約事項:

トリガーにニックネームを使用することはできません。

BEFORE トリガーの場合は、 ID 列以外の生成列の列名を、トリガー・アクションによって指定することはできません。したがって、生成される識別値は BEFORE トリガーに認識されます。

アトミック・トリガーを作成する際には、ステートメント終了文字に注意する必要があります。データベース・マネージャーは、デフォルトではステートメント終了マーカーを「;」と見なします。「;」以外の文字を使用するには、スクリプト内でステートメント終了文字を手動で編集して、アトミック・トリガーを作成しなければなりません。たとえば、「;」を「#」などの別の特殊文字で置き換えます。

次に、以下のいずれかを行う必要があります。

- コマンド・エディター (これはコマンド・センターに置き換わるものです) で選択したスクリプト・タブを使用して「ツール (tools)」—>「ツール設定 (tools settings)」メニューから区切り文字を変更し、スクリプトを実行します。または、
- コマンド行プロセッサーから、以下を使用します。

```
db2 -td <delimiter> -vf <script>
```

ここで、delimiter は代わりのステートメント終了文字で、 <script> はその中の新しい区切り文字を使って変更したスクリプトです。

手順:

コントロール・センターを使用してトリガーを作成するには、以下のようにします。

1. オブジェクト・ツリーを順に展開し、**「トリガー (Triggers)」**フォルダーを表示します。
2. **「トリガー (Triggers)」**フォルダーを右クリックして、ポップアップ・メニューから**「作成 (Create)」**を選択します。
3. トリガーに関する情報を指定します。
4. トリガーによって呼び出すアクションを指定し、**「OK」**をクリックします。

コマンド行を使用してトリガーを作成するには、以下のように入力します。

```
CREATE TRIGGER <name>
   <action> ON <table_name>
   <operation>
   <triggered_action>
```

次の SQL ステートメントは、新人が採用されるたびに従業員の数を増やすトリガーが作成されます。これによって、EMPLOYEE 表に行が追加されるたびに、COMPANY_STATS 表の従業員数 (NBEMP) 列に 1 が加算されます。

```
CREATE TRIGGER NEW_HIRED
   AFTER INSERT ON EMPLOYEE
   FOR EACH ROW
   UPDATE COMPANY_STATS SET NBEMP = NBEMP+1;
```

トリガー本体には、INSERT、探索 UPDATE、探索 DELETE、全選択、SET 変位変数、および SIGNAL SQLSTATE のうちの 1 つ以上の SQL ステートメントを含めることができます。トリガーは、それが参照する INSERT、 UPDATE、または DELETE ステートメントの前または後に起動できます。

関連概念:

- 125 ページの『トリガーの従属関係』
- 「*アプリケーション開発ガイド サーバー・アプリケーションのプログラミング*」の『INSERT、UPDATE、および DELETE トリガー』
- 「*アプリケーション開発ガイド サーバー・アプリケーションのプログラミング*」の『アプリケーション開発でのトリガー』
- 「*アプリケーション開発ガイド サーバー・アプリケーションのプログラミング*」の『トリガー作成のガイドライン』
- 125 ページの『トリガーを使用したビューの内容の更新』

関連タスク:

- 211 ページの『トリガーのドロップ』
- 「*アプリケーション開発ガイド サーバー・アプリケーションのプログラミング*」の『トリガーの作成』
- 「*アプリケーション開発ガイド サーバー・アプリケーションのプログラミング*」の『トリガーを使用した業務規則の定義』

- 「アプリケーション開発ガイド サーバー・アプリケーションのプログラミング」の『トリガーを使用したアクションの定義』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『CREATE TRIGGER ステートメント』

トリガーの従属関係

他のオブジェクトに対するトリガーの従属関係は、すべて SYSCAT.TRIGDEP カタログに記録されます。トリガーはさまざまなオブジェクトに従属する可能性があります。これらのオブジェクトと従属のトリガーについては、DROP ステートメントで詳細に説明されています。

これらのオブジェクトのどれかをドロップすると、トリガーは機能しなくなりますが、その定義はカタログ内に残ります。そのトリガーを再び有効にするには、カタログからその定義を取り出し、新しい CREATE TRIGGER ステートメントをサブミットしてください。

トリガーをドロップすると、その定義は SYSCAT.TRIGGERS カタログ・ビューから削除され、その従属関係もすべて SYSCAT.TRIGDEP カタログ・ビューから削除されます。トリガーに UPDATE、INSERT、または DELETE 従属性のあるパッケージは、すべて無効になります。

従属オブジェクトがビューで、それが作動不能になっている場合は、トリガーにも作動不能のマークが付けられます。作動不能のマークが付けられたトリガーに従属するパッケージがあれば、すべて無効にされます。

関連概念:

- 125 ページの『トリガーを使用したビューの内容の更新』

関連タスク:

- 123 ページの『トリガーの作成』
- 211 ページの『トリガーのドロップ』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『CREATE TRIGGER ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『DROP ステートメント』

トリガーを使用したビューの内容の更新

INSTEAD OF トリガーは、本来更新できないビューのために削除、挿入、または更新要求を行うために使用できます。このタイプのトリガーを利用するアプリケーションは、ビューが表であるかのように、ビューに対して更新操作を書き込むことができます。

たとえば、以下の SQL ステートメントを使用してビューを作成することができます。

```
CREATE VIEW EMPV(EMPNO, FIRSTNME, MIDINIT, LASTNAME, PHONENO, HIREDATE,
      DEPTNAME)
   AS SELECT EMPNO, FIRSTNME, MIDINIT, LASTNAME, PHONENO, HIREDATE, DEPTNAME
      FROM EMPLOYEE, DEPARTMENT WHERE EMPLOYEE.WORKDEPT = DEPARTMENT.DEPTNO
```

EMPV ビューの本体にある結合のために、以下のステートメントが追加されるまで、ビューを使用して基礎表でデータを更新することはできません。

```
CREATE TRIGGER EMPV_INSERT INSTEAD OF INSERT ON EMPV
   REFERENCING NEW AS NEWEMP DEFAULTS NULL FOR EACH ROW
   INSERT INTO EMPLOYEE (EMPNO, FIRSTNME, MIDINIT, LASTNAME, WORKDEPT,
      PHONENO, HIREDATE)
   VALUES(EMPNO, FIRSTNME, MIDINIT, LASTNAME,
       COALESCE((SELECT DEPTNO FROM DEPARTMENT AS D
       WHERE D.DEPTNAME = NEWEMP.DEPTNAME),
       RAISE_ERROR('70001', 'Unknown department name')),
       PHONENO, HIREDATE)
```

この CREATE TRIGGER ステートメントは、EMPV ビューに対する INSERT 要求が行われるようにします。

```
CREATE TRIGGER EMPV_DELETE INSTEAD OF DELETE ON EMPV
   REFERENCING OLD AS OLDEMP FOR EACH ROW
   DELETE FROM EMPLOYEE AS E WHERE E.EMPNO = OLDEMP.EMPNO
```

この CREATE TRIGGER ステートメントは、EMPV ビューに対する DELETE 要求が行われるようにします。

```
CREATE TRIGGER EMPV_UPDATE INSTEAD OF UPDATE ON EMPV
   REFERENCING NEW AS NEWEMP
               OLD AS OLDEMP
   DEFAULTS NULL FOR EACH ROW
   BEGIN ATOMIC
      VALUES(CASE WHEN NEWEMP.EMPNO = OLDEMP.EMPNO THEN 0
                  ELSE RAISE_ERROR('70002', 'Must not change EMPNO') END);
      UPDATE EMPLOYEE AS E
         SET (FIRSTNME, MIDINIT, LASTNAME, WORKDEPT, PHONENO, HIREDATE) =
             (NEWEMP.FIRSTNME, NEWEMP.MIDINIT, NEWEMP.LASTNAME,
              COALESCE((SELECT DEPTNO FROM DEPARTMENT AS D
                          WHERE D.DEPTNAME = NEWEMP.DEPTNAME),
              RAISE_ERROR('70001', 'Unknown department name')),
              NEWEMP.PHONENO, NEWEMP.HIREDATE)
         WHERE NEWEMP.EMPNO = E.EMPNO;
   END
```

この CREATE TRIGGER ステートメントは、EMPV ビューに対する UPDATE 要求が行われるようにします。

関連タスク:

- 123 ページの『トリガーの作成』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『CREATE TRIGGER ステートメント』

ユーザー定義関数 (UDF) またはメソッドの作成

ユーザー定義関数 (UDF) は、SQL の組み込み関数によるサポートを拡張および追加するものであり、組み込み関数が使用できるところであればどこでも使用することができます。 UDF を、以下のいずれかとして作成することができます。

- 外部関数。プログラム言語で作成されたもの。

- ソース関数。それを実現したものは、既存の他の関数から継承されます。

以下の 3 つのタイプの UDF があります。

スカラー
呼び出されるたびに、単一値の応答を戻します。たとえば、組み込み関数 SUBSTR() はスカラー関数です。スカラー UDF は、外部関数またはソース関数のいずれも可能です。

列 1 組みの類似値 (1 つの列) から、単一値の応答を戻します。 DB2® では、総計関数と呼ばれる場合もあります。列関数の 1 つの例は、組み込み関数 AVG() です。外部列 UDF を DB2 に定義することはできませんが、組み込み列関数の 1 つのソース関数となる列 UDF は定義することができます。これは、特殊タイプの場合に便利です。

たとえば、基本タイプ INTEGER で定義された特殊タイプ SHOESIZE がある場合、組み込み関数 AVG(INTEGER) のソース関数となる UDF AVG(SHOESIZE) を定義することができ、それは列関数になります。

表 それを参照する SQL ステートメントに対して 1 つの表を戻します。表関数は、SELECT ステートメントの FROM 文節の中でのみ参照できます。このような関数は、DB2 データでないデータのため、またはそのようなデータを DB2 表に変換するために、 SQL 言語処理能力を利用するために使用できます。

たとえば、表関数は、ファイルを取り出してそれを表に変換したり、ワールド・ワイド・ウェブ (WWW) からのサンプル・データを表の形にしたり、あるいは Lotus® Notes データベースにアクセスして、メール・メッセージの日付、送信者、テキストなどの情報を戻したりします。この情報は、データベース内の他の表と結合することができます。

表関数は、外部関数のみが可能です。ソース関数にはできません。

既存の UDF についての情報は、 SYSCAT.FUNCTIONS および SYSCAT.FUNCPARMS カタログ・ビューの中に記録されます。システム・カタログには UDF の実行可能コードは含まれません。 (このため、バックアップおよびリカバリーの計画を作成する場合には、 UDF 実行可能コードをどのように管理するかを考慮する必要があります。)

UDF のパフォーマンスに関する統計は、 SQL ステートメントをコンパイルするときに重要です。

関連概念:
- 「*SQL リファレンス 第 1 巻*」の『スカラー関数』
- 「*SQL リファレンス 第 1 巻*」の『ユーザー定義関数』
- 「*SQL リファレンス 第 1 巻*」の『表関数』
- 128 ページの『関数マッピングの作成』
- 「*管理ガイド: パフォーマンス*」の『ユーザー定義関数の統計』
- 「*管理ガイド: パフォーマンス*」の『手動でのカタログ統計更新の一般規則』
- 「*アプリケーション開発ガイド クライアント・アプリケーションのプログラミング*」の『DB2 ユーザー定義関数およびメソッド』

関連タスク:

- 129 ページの『関数テンプレートの作成』

関連資料:

- 「*SQL リファレンス 第 1 巻*」の『関数』
- 「*SQL リファレンス 第 2 巻*」の『CREATE FUNCTION ステートメント』

ユーザー定義関数 (UDF) またはメソッドの作成に関する詳細

このセクションでは、ユーザー定義の機能またはメソッドの作成時に必要な、総合的な考慮事項に関する情報を提供します。

関数マッピングの作成

フェデレーテッド・データベースでは、 1 つまたは複数のデータ・ソースを持つローカル関数またはローカル関数テンプレートをマップする必要があるときに、関数マッピングを作成します。多くのデータ・ソース関数で、デフォルトの関数マッピングが提供されています。

以下の場合に、関数マッピングが便利です。

- 新規の組み込み関数がデータ・ソースで使用可能になるとき。
- データ・ソースでのユーザー定義関数をローカル関数にマップする必要があるとき。
- アプリケーションで、デフォルト・マッピングで提供されるものとは異なるデフォルトの動作が求められるとき。

CREATE FUNCTION MAPPING ステートメントで定義された関数マッピングは、フェデレーテッド・データベースに保管されます。

関数 (関数テンプレート) には、データ・ソース関数と同数の入力パラメーターを指定する必要があります。さらに、フェデレートされる側の入力パラメーターのデータ・タイプと、データ・ソース側の入力パラメーターのデータ・タイプには、互換性がなければなりません。これらの要件は戻り値にも適用されます。

CREATE FUNCTION MAPPING ステートメントを、関数マッピングを作成するために使用します。たとえば、サーバー ORACLE1 で Oracle の AVGNEW 関数と DB2® の同等関数との関数マッピングを作成するには、次のようにします。

```
CREATE FUNCTION MAPPING ORAVGNEW FOR SYSIBM.AVG(INT) SERVER ORACLE1
OPTIONS (REMOTE_NAME 'AVGNEW')
```

このステートメントを使用するには、フェデレーテッド・データベースで SYSADM または DBADM 権限のいずれかを持っている必要があります。関数マッピングの属性は SYSCAT.FUNCMAPPINGS に保管されます。

フェデレーテッド・サーバーは入力ホスト変数をバインドしたり、LOB、 LONG VARCHAR/VARGRAPHIC、 DATALINK、特殊および構造型タイプの結果を検索したりしません。入力パラメーターまたは戻り値に上記のいずれかのタイプが含まれていると、関数マッピングは作成できません。

関連概念:

- 「*アプリケーション開発ガイド サーバー・アプリケーションのプログラミング*」の『Transform 関数を使用したホスト言語プログラムのマッピング』

関連タスク:

- 129 ページの『関数テンプレートの作成』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『CREATE FUNCTION MAPPING ステートメント』

関数テンプレートの作成

フェデレーテッド・システムでは、関数テンプレートによって関数マッピング用の「アンカー」が提供されます。アンカーは、該当する DB2 関数がフェデレーテッド・サーバーに存在しない場合にデータ・ソース関数のマッピングを可能にするために使用します。関数マッピングでは、関数テンプレートが存在するか、または DB2 に同様の関数が備わっていなければなりません。

テンプレートとは関数シェル、すなわち、名前、入力パラメーター、および戻り値にほかなりません。関数にはローカルの実行可能コードはありません。

制約事項:

関数用のローカル実行可能コードがないため、データ・ソースに使用可能な関数があっても、関数テンプレートの呼び出しが失敗するという可能性があります。たとえば、次のような照会を考えてみます。

```
SELECT myfunc(C1)
   FROM nick1
   WHERE C2 < 'A'
```

DB2 および nick1 が参照するオブジェクトの入ったデータ・ソースの照合順序が同じではない場合、関数がデータ・ソースにある間に比較を行う必要があるため、照会は失敗します。照合順序が同じであれば、比較操作は、 `myfunc` が参照する基礎関数を持つデータ・ソースで行うことができます。

関数 (関数テンプレート) には、データ・ソース関数と同数の入力パラメーターを指定する必要があります。フェデレートされる側の入力パラメーターのデータ・タイプと、データ・ソース側の入力パラメーターのデータ・タイプには、互換性がなければなりません。これらの要件は戻り値にも適用されます。

手順:

関数テンプレートは、 CREATE FUNCTION ステートメントに AS TEMPLATE キーワードを指定して作成します。テンプレートを作成したら、 CREATE FUNCTION MAPPING ステートメントを使ってテンプレートをデータ・ソースにマップします。

たとえば、サーバー S1 に関数 MYS1FUNC 用の関数テンプレートと関数マッピングを作成するには、次のようにします。

```
CREATE FUNCTION MYFUNC(INT) RETURNS INT AS TEMPLATE

CREATE FUNCTION MAPPING S1_MYFUNC FOR MYFUNC(INT) SERVER S1 OPTIONS
(REMOTE_NAME 'MYS1FUNC')
```

関連概念:

- 128 ページの『関数マッピングの作成』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『CREATE FUNCTION (ソースまたはテンプレート) ステートメント』

ユーザー定義タイプ (UDT)

ユーザー定義タイプ (UDT) は、データベース内でユーザーによって作成される名前付きデータ・タイプです。 UDT は、それぞれがタイプを持つ名前付き属性の順序がある組み込みデータ・タイプまたは構造型と、共通表示を共用する特殊タイプとなることができます。構造型はタイプ階層を定義しますが、別の構造型 (スーパータイプと呼ばれる) のサブタイプとなることができます。

UDT は、強い型指定をサポートしています。これは、UDT が他のタイプと同じ表示を共用していても、指定された UDT の値が同じタイプ階層の同じ UDT (複数も可) の値とのみ互換性のあるものと見なされるという意味です。

SYSCAT.DATATYPES カタログ・ビューを使用して、データベース用に定義された UDT を見ることができます。このカタログ・ビューでは、データベースの作成時にデータベース・マネージャーが定義したデータ・タイプも示しています。

UDT は、ほとんどのシステム提供関数 (組み込み関数) には引き数として使用することはできません。これらの演算子や他の演算子を使用するには、ユーザー定義関数を作成する必要があります。

以下の場合にのみ、UDT をドロップすることができます。

- 既存の表の列定義の中で使用されていない場合。
- 既存の型付き表または型付きビューのタイプとして使用されていない場合。
- ドロップできない UDF 関数の中で使用されていない場合。ある UDF にビュー、トリガー、表チェック制約、または別の UDF が従属している場合、その UDF はドロップできません。

UDT をドロップすると、それに従属する関数があれば、それもドロップされます。

関連概念:

- 132 ページの『ユーザー定義の構造化タイプの作成』

関連タスク:

- 131 ページの『ユーザー定義特殊タイプの作成』

関連資料:

- 「*SQL リファレンス 第 1 巻*」の『ユーザー定義タイプ』
- 「*SQL リファレンス 第 1 巻*」の『データ・タイプ』

ユーザー定義タイプ (UDT) の作成に関する詳細

特殊タイプおよび構造タイプの定義については、フェデレーテッド・タイプ・マッピングにあるとおり、以下のページで説明されます。

ユーザー定義特殊タイプの作成

ユーザー定義特殊タイプは、整数、10 進数、または文字タイプなど、既存のタイプから導出されたデータ・タイプです。特殊タイプは、CREATE DISTINCT TYPE ステートメントを使用して作成することができます。

制約事項:

インスタンスにあるように、 WITH COMPARISONS 文節が CREATE DISTINCT TYPE ステートメントで指定された場合には、同じ特殊タイプのインスタンスを相互に比較することができます。 WITH COMPARISONS 文節は、ソース・データがラージ・オブジェクト、DATALINK、LONG VARCHAR、または LONG VARGRAPHIC タイプである場合は、指定できません。

特殊タイプのインスタンスは、ソース・タイプに定義された関数の引き数または操作のオペランドとして使用することはできません。同様に、ソース・タイプは、特殊タイプを使用するよう定義された引き数またはオペランドの中で使用することはできません。

手順:

次の SQL ステートメントは、特殊タイプ t_educ を smallint として作成するものです。

```
CREATE DISTINCT TYPE T_EDUC AS SMALLINT WITH COMPARISONS
```

特殊タイプを作成したなら、それを使用して、 CREATE TABLE ステートメントに列を定義することができます。

```
CREATE TABLE EMPLOYEE
   (EMPNO      CHAR(6)      NOT NULL,
    FIRSTNME   VARCHAR(12)  NOT NULL,
    LASTNAME   VARCHAR(15)  NOT NULL,
    WORKDEPT   CHAR(3),
    PHONENO    CHAR(4),
    PHOTO      BLOB(10M)    NOT NULL,
    EDLEVEL    T_EDUC)
IN RESOURCE
```

特殊タイプを作成すると、特殊タイプとソース・タイプ間のキャストもサポートされるようになります。したがって、タイプ T_EDUC の値を SMALLINT 値にキャストすることができ、 SMALLINT 値を T_EDUC 値にキャストすることができます。

変換を利用して、UDT を基本データ・タイプに、基本データ・タイプを UDT に変換することができます。変換関数は CREATE TRANSFORM ステートメントによって作成します。

変換のサポートは、CREATE METHOD ステートメントおよび CREATE FUNCTION ステートメントの拡張形式の中にも見られます。

関連概念:

- 130 ページの『ユーザー定義タイプ (UDT)』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『CREATE DISTINCT TYPE ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『CREATE TRANSFORM ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『CREATE METHOD ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『CREATE FUNCTION (ソースまたはテンプレート) ステートメント』
- 「*SQL リファレンス 第 1 巻*」の『ユーザー定義タイプ』
- 「*SQL リファレンス 第 1 巻*」の『データ・タイプ』

ユーザー定義の構造化タイプの作成

構造化タイプとは、1 つ以上の名前属性を持つユーザー定義データ・タイプのことを言います。各属性には、名前と、それ自身のデータ・タイプがあります。属性は、タイプのインスタンスについて記述するプロパティーです。構造化タイプは表のタイプの部分にあたり、表の各列は構造化タイプの属性の 1 つから名前とデータ・タイプを導出します。

関連概念:

- 「*アプリケーション開発ガイド サーバー・アプリケーションのプログラミング*」の『ユーザー定義構造化型』
- 「*アプリケーション開発ガイド サーバー・アプリケーションのプログラミング*」の『構造化型階層』

関連タスク:

- 「*アプリケーション開発ガイド サーバー・アプリケーションのプログラミング*」の『構造化型の作成』
- 「*アプリケーション開発ガイド サーバー・アプリケーションのプログラミング*」の『構造化型階層の作成』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『CREATE TYPE (構造化) ステートメント』
- 「*SQL リファレンス 第 1 巻*」の『ユーザー定義タイプ』

タイプ・マッピングの作成

フェデレーテッド・システムでは、タイプ・マッピングにより、データ・ソース表内の特定データ・タイプと DB2 特殊データ・タイプのビューをマップできます。タイプ・マッピングは、1 つのデータ・ソース、またはデータ・ソースのある範囲 (タイプ、バージョン) に適用できます。

組み込みデータ・ソース・タイプおよび組み込み DB2 タイプには、デフォルトのデータ・タイプ・マッピングが用意されています。新規データ・タイプ・マッピング (ユーザーが作成するもの) は、 SYSCAT.TYPEMAPPINGS ビューにリストされます。

制約事項:

LOB、LONG VARCHAR/VARGRAPHIC、DATALINK、構造型、または特殊タイプ用のタイプ・マッピングを作成することはできません。

手順:

タイプ・マッピングは、CREATE TYPE MAPPING ステートメントを使って作成します。このステートメントを使用するには、フェデレーテッド・データベースで SYSADM または DBADM 権限のいずれかを持っている必要があります。

タイプ・マッピング・ステートメントの例は、次のとおりです。

```
CREATE TYPE MAPPING MY_ORACLE_DEC FROM SYSIBM.DECIMAL(10,2)
TO SERVER ORACLE1 TYPE NUMBER([10..38],2)
```

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『CREATE TYPE MAPPING ステートメント』
- 「*アプリケーション開発ガイド クライアント・アプリケーションのプログラミング*」の『DB2 と OLE DB の間のデータ・タイプ・マッピング』

ビューの作成

ビューは 1 つまたは複数の基本表、ニックネーム、またはビューから導出されるもので、データの検索時には基本表と交換可能なものとして使用されます。ビューの中に表示されるデータに変更が加えられると、表そのもののデータも変更されます。

ビューを作成することによって、重要データについてはアクセスを限定し、その他のデータについては一般にアクセスを許可することができます。

ビュー定義の SELECT リストに、基本表の ID 列の名前が直接または間接的に含まれている場合、そのビューに挿入を実行するときには、 INSERT ステートメントが基本表の ID 列を直接参照している場合と同じ規則が適用されます。

上記のようなビューの使用に加えて、次のような目的でビューを使用することができます。

- アプリケーション・プログラムに影響を及ぼさずに表を変更する。このことは、基本表に基づいてビューを作成することによって生じます。基本表を使用するアプリケーションは、新しいビューの作成によって影響を受けることはありません。新しいアプリケーションは、基本表を使用するアプリケーションではなく、異なる目的のために作成されたビューを使用できます。
- 列の中の値を合計する、最大値を選択する、または、それらの値を平均する。
- 1 つまたは複数のデータ・ソースの中の情報へのアクセスを提供する。 CREATE VIEW ステートメント内のニックネームを参照し、複数ロケーション/グローバル・ビュー (ビューは異なるシステム上にある複数データ・ソース内の情報を結合できる) を作成することができます。

  標準の CREATE VIEW 構文を使ってニックネームを参照するビューを作成すると、基礎オブジェクトまたはデータ・ソースのオブジェクトへのアクセスにビュ

ー作成者認証 ID ではなくビュー・ユーザーの認証 ID が使用されるという事実に注意を促す警告が表示されます。この警告を表示しないようにするには、FEDERATED キーワードを使用します。

ビューを作成する代わりにネストした表式または共通表式を使うこともできます。その場合、カタログ参照が少なくなり、パフォーマンスは高くなります。

前提条件:

基本表、ニックネーム、またはビューの基礎となるビューが、ビューを作成する前にすでに存在していなければなりません。

制約事項:

定義の中で UDF を使うビューを作成することもできます。ただし、このビューを最新の関数が含まれるように更新するためには、そのビューをドロップしてから再作成しなければなりません。ビューが UDF に依存している場合、その関数はドロップできません。

次の SQL ステートメントは、定義内に関数の含まれるビューを作成するものです。

```
CREATE VIEW EMPLOYEE_PENSION (NAME, PENSION)
  AS SELECT NAME, PENSION(HIREDATE,BIRTHDATE,SALARY,BONUS)
  FROM EMPLOYEE
```

UDF 関数の PENSION によって、従業員が現時点で受け取ることのできる年金が計算されます。その計算には、HIREDATE、BIRTHDATE、SALARY、および BONUS が使われます。

手順:

コントロール・センターを使用してビューを作成するには、以下のようにします。

1. オブジェクト・ツリーを順に展開し、**「ビュー (Views)」**フォルダーを表示します。
2. **「ビュー (Views)」**フォルダーを右クリックして、ポップアップ・メニューから**「作成 (Create)」**を選択します。
3. 情報をすべて入力し、**「OK」**をクリックします。

コマンド行を使用してビューを作成するには、以下のように入力します。

```
CREATE VIEW <name> (<column>, <column>, <column>)
   SELECT <column_names> FROM <table_name>
   WITH CHECK OPTION
```

たとえば、EMPLOYEE 表には給与に関する情報が入っているかもしれませんが、そうした情報は誰もが利用できるようにすべきではありません。しかし、従業員の電話番号はだれでもアクセスできるようにすべきです。このような場合は、LASTNAME 列と PHONENO 列だけからなるビューを作成できます。ビューへのアクセスは PUBLIC に与えるようにし、EMPLOYEE 表全体へのアクセスは、給与情報を見る権限のある人だけに制限するようにします。

ビューを使うと、表データのうちアプリケーション・プログラムで利用できるサブセットを作成し、挿入または更新するデータの妥当性検査を実行することができます。ビューの列名は、元の表の対応する列名と違うものにすることができます。

ビューを使うと、プログラムやエンド・ユーザー照会で表データを見る方法の点で柔軟性が高くなります。

以下の SQL ステートメントは、EMPLOYEE 表に 1 つのビューを作成します。このビューは部門 A00 のすべての従業員を、従業員番号と電話番号の情報とともにリストします。

```
CREATE VIEW EMP_VIEW (DA00NAME, DA00NUM, PHONENO)
   AS SELECT LASTNAME, EMPNO, PHONENO FROM EMPLOYEE
   WHERE WORKDEPT = 'A00'
   WITH CHECK OPTION
```

このステートメントの最初の行では、ビューの名前を指定し、その列を定義しています。 EMP_VIEW という名前は、 SYSCAT.TABLES の中のそのスキーマ内でユニークなものでなければなりません。ビュー名は表名の 1 つとして表示されますが、データは含まれていません。このビューの列は DA00NAME、 DA00NUM、および PHONENO の 3 つであり、それぞれ、 EMPLOYEE 表の LASTNAME、EMPNO、および PHONENO の列に対応するものです。最初の行に指定する列名は、SELECT ステートメントの選択リストに 1 対 1 に対応します。列名を指定しないと、ビューの列名には SELECT ステートメントの結果表の列と同じ名前が使われます。

第 2 行は、データベースから選択する値について記述する SELECT ステートメントです。これには、ALL、DISTINCT、FROM、WHERE、GROUP BY、および HAVING という文節を含めることができます。ビューのための列を選択する元のデータ・オブジェクトの名前を、 FROM 文節の後に指定します。

WITH CHECK OPTION 文節は、ビューに対して更新する行または挿入する行をビュー定義に照らしてチェックし、定義に従っていない場合にはリジェクトすることを指定するものです。これによってデータ保全性は向上しますが、余分な処理が必要になります。この文節を省略すると、挿入する行または更新する行がビュー定義に照らしてチェックされることはありません。

次の SQL ステートメントは、EMPLOYEE 表に基づく同じビューを、 SELECT AS 文節を使って作成するものです。

```
CREATE VIEW EMP_VIEW
      SELECT LASTNAME AS DA00NAME,
             EMPNO AS DA00NUM,
             PHONENO
      FROM EMPLOYEE
      WHERE WORKDEPT = 'A00'
      WITH CHECK OPTION
```

関連概念:

- 「*SQL リファレンス 第 1 巻*」の『ビュー』
- 256 ページの『表およびビューの特権』
- 267 ページの『ビューを使用したデータ・アクセスの制御』
- 125 ページの『トリガーを使用したビューの内容の更新』

関連タスク:

- 136 ページの『型付きビューの作成』
- 189 ページの『表またはビューからの行の除去』
- 213 ページの『ビューの変更またはドロップ』
- 215 ページの『作動不能ビューの回復』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『CREATE VIEW ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『INSERT ステートメント』

ビューの作成に関する詳細

型付きビューは、事前定義された構造タイプに基づいています。

型付きビューの作成

手順:

CREATE VIEW ステートメントを使用して型付きビューを作成できます。

関連概念:

- 「*アプリケーション開発ガイド サーバー・アプリケーションのプログラミング*」の『型付きビュー』

関連タスク:

- 「*アプリケーション開発ガイド サーバー・アプリケーションのプログラミング*」の『型付きビューの作成』
- 「*アプリケーション開発ガイド サーバー・アプリケーションのプログラミング*」の『型付きビューの変更』
- 「*アプリケーション開発ガイド サーバー・アプリケーションのプログラミング*」の『型付きビューのドロップ』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『CREATE VIEW ステートメント』

マテリアライズ照会表の作成

マテリアライズ照会表 は、その定義が照会の結果に基づいている表です。したがって、通常、マテリアライズ照会表には、その定義の基礎となる表 (複数も可) に存在するデータに基づいた、事前に計算された結果が含まれています。 SQL コンパイラーが、基本表に対するよりも、マテリアライズ照会表に対する照会のほうが効果的に実行できると判断する場合は、照会はマテリアライズ照会表に対して実行され、基本表に対する照会よりも速く結果を得られます。

制約事項:

REFRESH DEFERRED を使って定義したマテリアライズ照会表は、静的 SQL を最適化するためには使用されません。

CURRENT REFRESH AGE 特殊レジスターをゼロ以外の値に設定するに際しては、注意が必要です。照会の処理を最適化するために、基礎となる基本表の値を表さないマテリアライズ照会表を使用できるようにすると、照会の結果は基礎表のデータを正確には表しません。基本表のデータが変更されていないことを知っている、またはデータについての知識に基づいて結果のエラーの程度を受け入れるのであれば、これは無理のない設定といえるでしょう。

有効な *fullselect* に基づく新しい基本表を作成したい場合は、表の作成時に DEFINITION ONLY キーワードを指定します。表作成操作が完了すると、新しい表は基本表としてではなくマテリアライズ照会表として扱われます。たとえば、次のようにして LOAD と SET INTEGRITY で使用される例外表を作成できます。

```
CREATE TABLE XT AS
   (SELECT T.*, CURRENT TIMESTAMP AS TIMESTAMP,CLOB(",32K)
   AS MSG FROM T) DEFINITION ONLY
```

以下は、マテリアライズ照会表に関連する重要な制約事項です。

1. マテリアライズ照会表は変更できない。
2. 基本表にマテリアライズ照会表がある場合、基本表の列の長さは変更できない。
3. マテリアライズ照会表にデータをインポートできない。
4. マテリアライズ照会表にユニーク索引を作成できない。
5. 1 つまたは複数のニックネームを参照する結果に基づくマテリアライズ照会表は作成できない。

手順:

パーティション・データベース環境のすべてのノードにわたって表を複製するために、複製オプションを使用してマテリアライズ照会表を作成できます。これらは、「複製マテリアライズ照会表」と呼ばれています。

マテリアライズ照会表または複製マテリアライズ照会表の分離レベルが照会の分離レベルと等しいか、同レベルを上回る場合、照会の最適化には普通、マテリアライズ照会表または複製マテリアライズ照会表が使用されます。たとえば、照会がカーソル固定 (CS) 分離レベル下で実行される場合、最適化には CS または上位の分離レベルで定義されたマテリアライズ照会表と複製マテリアライズ照会表だけが使用されます。

マテリアライズ照会表を作成するには、 AS *fullselect* 文節と IMMEDIATE または REFRESH DEFERRED オプションを指定した、 CREATE TABLE ステートメントを使用します。

マテリアライズ照会表の列の名前は、一意的に識別することができます。列名のリストには、全選択の結果表の列と同じ数だけ名前がなければなりません。全選択の結果表に重複列があったり、無名列があったりする場合には、列名のリストを指定することが必要です。無名列は、選択リストの AS 文節に名前が指定されていない定数、関数、式、またはセット演算のために生じます。列名のリストを指定しないと、表の列は、全選択の結果セットの列の名前を継承します。

マテリアライズ照会表を作成する時、システムがマテリアライズ照会表を保守するか、ユーザーがマテリアライズ照会表を保守するかを指定するオプションがあります。デフォルトは、システム保守で、それは MAINTAINED BY SYSTEM 文節を使

用して明示的に指定することができます。ユーザー保守によるマテリアライズ照会表は、MAINTAINED BY USER 文節を使用して指定します。

システム保守によるマテリアライズ照会表を作成した場合、基本表の変更時にマテリアライズ照会表を自動的にリフレッシュするかどうか、または REFRESH TABLE ステートメントを使ってマテリアライズ照会表をリフレッシュするかどうかを指定する追加オプションがあります。基本表 (複数も可) に変更が加えられたときにマテリアライズ照会表を自動的にリフレッシュするには、 REFRESH IMMEDIATE キーワードを指定します。以下の場合には、即時リフレッシュが便利です。

- 照会時、アクセスするデータが最新であるかを確認する必要がある
- 基本表の変更頻度が低い
- リフレッシュに多くの費用がかからない

このような状況の場合、マテリアライズ照会表は事前計算した結果を提供できます。マテリアライズ照会表のリフレッシュを据え置きたければ、REFRESH DEFERRED キーワードを指定します。 REFRESH DEFERRED を使用して指定されたマテリアライズ照会表は、基本表に対する変更を反映**しません**。マテリアライズ照会表は使用するべきですが、必ずそうしなければならないという意味ではありません。たとえば、DSS の照会を実行する場合、既存のデータが入っているマテリアライズ照会表を使用します。

マテリアライズ照会表が次の条件を満たす場合は、照会の代わりに REFRESH DEFERRED を使用して定義されたマテリアライズ照会表が使用されます。

- 次の場合以外は、即時リフレッシュ・サマリー表の全選択の制限に準拠している。
  - COUNT(*) または COUNT_BIG(*) を組み込むために SELECT リストが必要ではない。
  - SELECT リストには、MAX および MIN 列関数を組み込むことができる。
  - HAVING 文節を指定できる。

CURRENT REFRESH AGE 特殊レジスターを使用して、 REFRESH DEFERRED で定義されるマテリアライズ照会表が、リフレッシュが必要になる前に動的照会に使用できる時間を指定します。 CURRENT REFRESH AGE 特殊レジスターの値を設定するには、 SET CURRENT REFRESH AGE ステートメントを使用します。

CURRENT REFRESH AGE 特殊レジスターは ANY に設定することもでき、99999999999999 の値に設定すると、据え置きマテリアライズ照会を動的照会で使用できるようになります。この 9 の連続した値は、この特殊レジスターで指定できる最大値であり、 DECIMAL(20,6) というデータ・タイプのタイム・スタンプ期間値です。値ゼロ (0) は、 REFRESH IMMEDIATE で定義されたマテリアライズ照会表だけを使用して照会の処理を最適化できるということを表します。その場合には、REFRESH DEFERRED を使って定義したマテリアライズ照会表は、最適化に使用されません。

REFRESH IMMEDIATE を使って定義したマテリアライズ照会表は、静的照会にも動的照会にも適用可能で、 CURRENT REFRESH AGE 特殊レジスターを使用する必要はありません。

表が ENABLE QUERY OPTIMIZATION 文節を使用して定義されている時には、マテリアライズ照会表は、それに経路指定された照会を持っています。据え置きマテリアライズ照会表の場合には、 CURRENT REFRESH AGE 特殊レジスターが ANY に設定されています。しかしながら、ユーザー保守によるマテリアライズ照会表においては、 CURRENT REFRESH AGE 特殊レジスターの使用は、照会の経路再指定を制御する上で最善の方式とはいえません。 CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION 特殊レジスターは、どのようなキャッシュ・データが経路指定に使用可能かを示します。

ソース・データに影響する活動については、時間とともにマテリアライズ照会表には正確なデータが含まれなくなります。 REFRESH TABLE ステートメントを使用する必要があるでしょう。

関連概念:

- 「*SQL リファレンス 第 1 巻*」の『分離レベル』

関連タスク:

- 204 ページの『マテリアライズ照会表のプロパティーの変更』
- 205 ページの『マテリアライズ照会表のデータのリフレッシュ』
- 216 ページの『マテリアライズ照会またはステージング表のドロップ』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『CREATE TABLE ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『REFRESH TABLE ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『SET CURRENT REFRESH AGE ステートメント』
- 「*SQL リファレンス 第 1 巻*」の『CURRENT REFRESH AGE 特殊レジスター』
- 「*SQL リファレンス 第 1 巻*」の『CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION 特殊レジスター』
- 「*SQL リファレンス 第 2 巻*」の『SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION ステートメント』

ユーザー保守のマテリアライズ照会表の作成

ユーザー保守のマテリアライズ照会表 (MQT) は、サマリー・データの表がすでに存在するデータベース・システムで役立ちます。そのようなサマリー表を保守するカスタム・アプリケーションは、多く見られます。既存のサマリー表をユーザー保守の MQT として識別すると、照会オプティマイザーは基本表に対する照会の結果セットを計算するために既存のサマリー表を使用します。

注: 照会オプティマイザーは、静的 SQL 照会のアクセス・プランを選択するときに、ユーザー保守の MQT を使用しません。

制約事項:

ユーザー保守によるマテリアライズ照会表を作成する場合には、システム保守によるマテリアライズ照会表に関連した制約事項がそのまま適用されますが、以下の例外があります。

- INSERT、UPDATE、および DELETE 操作は、マテリアライズ照会表では許可されます。ただし、妥当性検査は、基礎となる基本表に対して行われません。データの正確さに関しては、ユーザーに責任があります。
- LOAD、EXPORT、IMPORT、およびデータ複製は、妥当性検査がないことを除いては、このタイプのマテリアライズ照会表において行われます。
- このタイプのマテリアライズ照会表では、REFRESH TABLE ステートメントを使用することは許可されません。
- このタイプのマテリアライズ照会表では、 SET INTEGRITY ... IMMEDIATE CHECKED ステートメントを使用することは許可されません。
- ユーザー保守によるマテリアライズ照会表は、 REFRESH DEFERRED として定義する必要があります。

その他の制約事項については、『マテリアライズ照会表の作成』トピックを参照してください。

手順:

マテリアライズ照会表を作成するには、 AS *fullselect* 文節と IMMEDIATE または REFRESH DEFERRED オプションを指定した、 CREATE TABLE ステートメントを使用します。

マテリアライズ照会表を作成する時、システムがマテリアライズ照会表を保守するか、ユーザーがマテリアライズ照会表を保守するかを指定するオプションがあります。デフォルトは、システム保守で、それは MAINTAINED BY SYSTEM 文節を使用して明示的に指定することができます。ユーザー保守によるマテリアライズ照会表は、MAINTAINED BY USER 文節を使用して指定します。

大規模なデータベース環境、またはデータウェアハウス環境では、ユーザー保守によるマテリアライズ照会表を保守およびロードするカスタム・アプリケーションがよくあります。

注: オプティマイザーがユーザー保守の MQT を検討するためには、照会最適化レベルをレベル 2、または 5 以上のレベルに設定する必要があります。

関連タスク:

- 141 ページの『ユーザー保守のマテリアライズ照会表にデータを追加する』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『CREATE TABLE ステートメント』

ユーザー保守のマテリアライズ照会表にデータを追加する

サマリー情報を保管するための表を作成した後、オプティマイザーが結果セットを判別するために使用するサマリー・データを、マテリアライズ照会表 (MQT) に追加することができます。

前提条件:

サマリー情報を保管するための表が存在することを確認してください。

手順:

ユーザー保守の MQT にデータを追加するには、トリガー、挿入操作、または LOAD、IMPORT、および DB2 DataPropagator ユーティリティーを使用できます。ユーザー保守の MQT に最初にデータを取り込むとき、LOAD または IMPORT ユーティリティーを使用すると、ロギングのオーバーヘッドを防止できます。

以下のステップは、ユーザー保守の MQT にデータを追加する標準的な方法を示しています。

- 新規レコードの作成や既存レコードの変更を防止するために、基本表を読み取り専用にします。
- 必要なデータを基本表から抽出して、それを外部ファイルに書き出します。
- 外部ファイルから MQT にデータをインポートまたはロードします。 CHECK PENDING NO ACCESS 状態の表には、LOAD または IMPORT ユーティリティーを使用できます。

  注: MQT に SQL 挿入操作を使用してデータを追加したい場合、 PENDING NO ACCESS 状態をリセットする必要があります。ただし、まず ALTER TABLE ステートメントの SET MATERIALIZED QUERY 文節に DISABLE QUERY OPTIMIZATION オプションを指定して使用することによりオプティマイザーを使用不可にして、データの確立が完了する前に動的 SQL 照会が間違ってこの MQT に最適化しないようにする必要があります。 MQT にデータが追加された後に、ALTER TABLE ステートメントの SET MATERIALIZED QUERY 文節に ENABLE QUERY OPTIMIZATION オプションを指定して使用することにより、最適化を使用可能にする必要があります。
- SQL 照会を新しい MQT に対して発行するには、CHECK PENDING NO ACCESS 状態をリセットします。このようにして、マテリアライズ・ビューのデータ保全性の責任を自分が負うことを示します。これを行うためのステートメントは、次のとおりです。

  ```
  DB2 SET INTEGRITY FOR example ALL IMMEDIATE UNCHECKED
  ```

- 基本表を read/write にしてください。

注: オプティマイザーがユーザー保守の MQT を検討するためには、照会最適化レベルをレベル 2、または 5 以上のレベルに設定する必要があります。

関連資料:

- 「コマンド・リファレンス」の『IMPORT コマンド』
- 「コマンド・リファレンス」の『LOAD コマンド』

ステージング表の作成

ステージング表 は、据え置きマテリアライズ照会表の増分保守サポートを可能にします。ステージング表は、マテリアライズ照会表を基礎表の内容で同期するためにマテリアライズ照会表に適用する必要がある変更を収集します。ステージング表を使用するなら、マテリアライズ照会表の即時リフレッシュが要求された時に、即時保守内容により引き起こされる高いロック競合を除去します。さらに、マテリアライズ照会表は、もはや REFRESH TABLE が実行される時ごとに全体を再生成する必要はありません。

マテリアライズ照会表は、複雑な照会をする時の応答時間を向上させる強力な手段であり、とりわけ以下の操作のいくつかを必要とする照会に威力を発揮します。

- 1 ディメンション以上の集約データ
- 表のグループの結合データおよび集約データ
- 通常アクセスされるデータのサブセットからのデータ
- パーティション・データベース環境での表からの再パーティション・データ、または表の一部

制約事項:

以下は、ステージング表に関連する重要な制約事項です。

1. ステージング表を定義するために使用される照会は、徐々に保守していけるものでなければなりません。すなわちその照会は、即時リフレッシュ・オプションを持つマテリアライズ照会表と同じ規則に従う必要があります。
2. 据え置きリフレッシュだけが、サポートするステージング表を持つことができます。さらに照会は、そのステージング表に関連したマテリアライズ照会表を定義します。マテリアライズ照会表は、REFRESH DEFERRED で定義しなければなりません。
3. ステージング表を使用してリフレッシュする時は、現時点のリフレッシュのみがサポートされます。

手順:

不整合、不完全、またはペンディング状態のステージング表は、いくつかの別の操作が発生するのでないかぎり、マテリアライズ照会表を徐々にリフレッシュするために使用することはできません。それらの操作は、ステージング表の内容を、それに関連したマテリアライズ照会表およびその基礎表と整合させ、ステージング表をペンディング状態から解除します。マテリアライズ照会表のリフレッシュに続き、そのステージング表の内容はクリアされ、ステージング表は通常の状態に設定されます。ステージング表は、SET INTEGRITY ステートメントにふさわしいオプションを付けて使用することにより、意図的に整理することもできます。整理すると、そのステージング表は不整合状態に変更されます。たとえば、以下のステートメントを実行すると、STAGTAB1 というステージング表を強制的に整理します。

```
SET INTEGRITY FOR STAGTAB1 PRUNE;
```

ステージング表が作成される時、それはペンディング状態に置かれ、さらに表が基礎表および関連したマテリアライズ照会表の内容に関して不整合または不完全であることを示す標識を持ちます。そのステージング表は、その基礎表から変更内容の収集を開始するために、ペンディングおよび不整合状態から解除される必要があり

ます。ペンディング状態にある時には、ステージング表のどの基礎表に対しても修正しようとすると失敗し、関連したマテリアライズ照会表をリフレッシュしようとしても失敗します。

ステージング表のペンディング状態を解除するには、いくつかの方法があります。

- SET INTEGRITY FOR <ステージング表名> STAGING IMMEDIATE UNCHECKED
- SET INTEGRITY FOR <ステージング表> IMMEDIATE CHECKED

関連タスク:

- 136 ページの『マテリアライズ照会表の作成』
- 204 ページの『マテリアライズ照会表のプロパティーの変更』
- 205 ページの『マテリアライズ照会表のデータのリフレッシュ』
- 216 ページの『マテリアライズ照会またはステージング表のドロップ』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『SET INTEGRITY ステートメント』

別名の作成

別名は、表、ニックネーム、またはビューを間接的に参照して、 SQL ステートメントを表やビューの修飾名とは無関係なものにするための手段です。表名やビュー名を変更しても、別名の定義を変えるだけで済みます。別名は他の別名に対して作成することもできます。別名は、ビューやトリガーの定義、また SQL ステートメントの中で使用できます。ただし、既存の表名やビュー名を参照する表チェック制約では使用できません。

前提条件:

別名は、定義時に存在していない表、ビュー、または別名に対しても定義できます。しかし、別名の含まれる SQL ステートメントのコンパイル時には、存在していなければなりません。

制約事項:

別名は既存の表名が使用できる所ならどこにでも使用できます。また、別名のチェーンに循環参照または反復参照がない限り他の別名を参照することもできます。

既存の表、ビュー、別名と同じ別名を作成することはできません。また、別名は同じデータベース内の表しか参照できません。 CREATE TABLE ステートメントまたは CREATE VIEW ステートメントでは、同じスキーマ内の別名と同じ表名やビュー名は使用できません。

別名の作成には特別な権限は必要ありません。ただし、自分の現在の許可 ID が所有するスキーマ以外のスキーマに別名を作成する場合は、 DBADM 権限が必要です。

別名または別名が参照するオブジェクトがドロップされると、その別名に依存するすべてのパッケージは無効のマークが付けられ、その別名に依存するすべてのビューおよびトリガーは作動不能のマークが付けられます。

手順:

コントロール・センターを使用して別名を作成するには、以下のようにします。

1. オブジェクト・ツリーを順に展開し、**「別名 (Aliases)」**フォルダーを表示します。
2. **「別名 (Aliases)」**フォルダーを右クリックして、ポップアップ・メニューから**「作成 (Create)」**を選択します。
3. 情報をすべて入力し、**「OK」**をクリックします。

コマンド行を使用して別名を作成するには、以下のように入力します。

```
CREATE ALIAS <alias_name> FOR <table_name>
```

別名は、ステートメントのコンパイル時に表名やビュー名に置き換えられます。別名または別名のチェーンが表名やビュー名に置換できないと、エラーになります。たとえば、WORKERS を EMPLOYEE の別名にした場合、コンパイル時には、

```
SELECT * FROM WORKERS
```

は、実際には次のものになります。

```
SELECT * FROM EMPLOYEE
```

次の SQL ステートメントは、EMPLOYEE 表に WORKERS という別名を作成するものです。

```
CREATE ALIAS WORKERS FOR EMPLOYEE
```

注: DB2 for OS/390 and z/Series は、ALIAS と SYNONYM という、別名についての 2 つの異なる概念を採用しています。これらの 2 つの概念は、DB2 Universal Database と以下の点で異なります。

- DB2 for OS/390 and z/Series の ALIAS
  - 作成者が特殊権限または特権を有していなければならない。
  - 他の別名を参照できない。
- DB2 for OS/390 and z/Series の SYNONYM
  - 作成者だけしか使用できない。
  - 常に修飾なしである。
  - 参照テーブルがドロップされると、ドロップされる。
  - ネーム・スペースを表またはビューと共用しない。

関連概念:

- 「*SQL リファレンス 第 1 巻*」の『別名』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『CREATE ALIAS ステートメント』

索引、索引の拡張、または索引の指定

索引は、行の位置のリストを、指定した 1 つまたは複数の列の内容によってソートしたものです。索引は、通常、表へのアクセスを高速にするために使用されます。しかし索引は、論理データ設計の点でも役立ちます。たとえば、*ユニーク索引* を使うと、列に重複値を入力できなくなるため、表内に同じ行ができることはありません。また、列の値を昇順にするか降順にするかを指定するためにも、索引を作成することができます。

索引の拡張とは、構造型または特殊タイプの列の索引と一緒に使用される索引オブジェクトです。

索引の指定はメタデータ構成です。それは、ニックネームが参照するデータ・ソース・オブジェクト (表またはビュー) の索引があることを、オプティマイザーに通知します。索引の指定は行位置のリストを含んでおらず、索引を記述したものにすぎません。オプティマイザーは索引の指定を使用して、ニックネームが示すオブジェクトへのアクセスを向上させます。 ニックネームが初めて作成される場合、 DB2® が認識できる形式の基本表用の索引がデータ・ソースにあれば、索引の指定が生成されます。

注: ビューが 1 つの表に対するものであれば、必要に応じて表のニックネームやビューのニックネームに対する索引の指定を作成します。

次の場合は、索引または索引の指定を手動で作成します。

- その結果として、パフォーマンスが向上する場合。たとえば、オプティマイザーで、ネストされたループ結合の内部表として特定の表やニックネームを使うようにする場合、索引が存在しなければ、結合列に対する索引の指定を作成します。
- 基本表の索引が、その表のニックネームが作成された後に追加された場合。

索引の指定は、基本表の索引が存在しない場合に作成できます (DB2 は、 CREATE INDEX ステートメントの発行時にリモート索引があるかどうかを調べません)。また、UNIQUE キーワードを指定したときでも行をユニークなものにする必要はありません。

DB2 索引アドバイザーは、最適の索引セットを選ぶのに役立つウィザードです。このウィザードは、コントロール・センターからアクセスできるウィザードです。互換性のあるユーティリティーとして、*db2advis* があります。

索引は、列ごとに基本表の中に定義されます。索引は、表の作成者、または特定の列では直接アクセスが必要であることについて知っているユーザーが作成できます。 1 次索引のキーは、ユーザー定義の索引がすでに存在しているのでない限り、主キーに対して自動的に作成されます。

1 つの基本表に対して、索引はいくつでも作成でき、そのようにして照会のパフォーマンスを高めることができます。しかし、索引の数が多ければ多いほど、更新、削除、挿入の操作時にデータベース・マネージャーの実行する修正作業は多くなります。更新事項の多い表に対してたくさんの索引を作成すると、要求の処理が遅くなってしまう可能性があります。したがって、索引の使用は、頻繁にアクセスするための利点があるということが明らかな場合だけにしてください。

1 つの索引あたりの列の最大数は 16 です。型付き表の索引作成をする場合、列の最大数は 15 です。索引キーの最大長は 1024 バイトです。前述のように、 1 つの表に対する索引キーが多くなると、要求の処理速度が低下することがあります。同様に、索引キーが大きくなっても、要求の処理速度が低下することがあります。

索引キー は 1 つの列または複数の列の集合のことであり、そこに索引が定義され、索引の有効性が判別されます。索引キーを構成する列の順序は、索引キーの作成に影響を与えることはありませんが、索引を使用するかどうかを決定するときに、オプティマイザーに影響を与えます。

索引を作成する表が空であっても、索引は作成されますが、表がロードされるか行が挿入される時点まで索引項目は作成されません。表が空でない場合、CREATE INDEX ステートメントの処理中に索引項目が作成されます。

クラスタリング索引 では、新しい行がキー値の近い既存の行と物理的に近い位置に挿入されます。このことによって、データ・ページへのアクセス・パターンの線形化が進み、プリフェッチの効果性が上がるので、照会の間のパフォーマンスが向上します。

主キーの索引をクラスタリング索引にしたい場合は、 CREATE TABLE で主キーを指定しないでください。主キーが作成されると、関連する索引を変更することはできなくなります。主キー文節を指定しないで CREATE TABLE を実行します。その後、クラスタリング属性を指定して CREATE INDEX を発行します。最後に、ALTER TABLE ステートメントを使用して、作成されたばかりの索引に対応する主キーを追加します。この索引が、主キーの索引として使用されます。

一般的に、クラスタリング索引が一意的なものであれば、クラスタリングの保守はより効果的に行えます。

ユニーク索引キーの一部ではないものの、その索引で保管され保守される列データは、*組み込み* 列と呼ばれます。組み込み列を指定できるのは、ユニーク索引についてだけです。組み込み列のある索引を作成しているときは、ユニーク・キー列だけが保管され、ユニークなものであると見なされます。組み込み列を使用すると、索引へのアクセスが関係しているときには、データ検索のパフォーマンスが向上します。

データベース・マネージャーは、最下レベルがリーフ・ノードで構成される場合の索引の保管に B+ 木構造を使用します。リーフ・ノードやリーフ・ページとは、実際の索引キー値が保管される場所です。索引の作成時には、上記の索引リーフ・ページを使用可能にしてオンラインでマージすることができます。オンラインでの索引デフラグは、大幅な削除および更新活動の後、多数の索引リーフ・ページにわずかの索引キーだけが残されるという状況を避けるために使用します。そのような状況でオンライン索引デフラグを行わなければ、データの再編成 (索引を含む場合と含まない場合とがある) によってスペースを再利用できるだけです。作成する索引に、索引ページをオンラインでデフラグを実行する機能が必要かどうかを判断するには、次の質問を考慮してください。すなわち、キーを物理的にリーフ・ページから削除するたびにマージ可能なスペースの有無をチェックするという追加のパフォーマンス上の費用、および (スペースが十分にある場合に) マージを完了するための

実費用とを費やすことは、索引用スペースの使用効率の改善という利点に勝るか、またスペースを再利用するために再編成を実行するという、必要性の小さい作業に見合う価値があるか、ということです。

注:

1. オンライン索引デフラグ後に解放されたページは、同じ表内の他の索引に再利用することだけに使用できます。全再編成を使用すると、解放されたページを他のオブジェクト (データベース管理ストレージでの作業時)、またはディスク・スペース (システム管理ストレージでの作業時) に使用できます。また、オンライン索引デフラグでは、索引のノンリーフ・ページは解放されませんが、全再編成では、索引を可能な限り最小化して、ノンリーフ・ページとリーフ・ページに加えて索引のレベル数も削減されます。
2. バージョン 8 よりも前で作成された索引では、表行の削除または更新の一部として、キーがリーフ・ページから物理的に削除されます。タイプ 2 索引では、行の削除または更新時にキーが削除済みとしてマークされるだけです。削除または更新がコミットされてしばらくしてからクリーンアップがなされるまで、キーはページから物理的には除去されません。そうしたクリーンアップは、キーが削除済みとしてマークされたページを変更する後続のトランザクションによりなされるかもしれません。クリーンアップは、REORG INDEXES ユーティリティーの CLEANUP ONLY [ALL | PAGES] オプションを使用して、明示的に起動することができます。

1 つのパーティション・データベース内の表に対する索引は、同じ CREATE INDEX ステートメントを使用して作成されます。これらの索引は、その表のパーティション・キーに基づいてパーティション化されます。表上の索引は、データベース・パーティション・グループ内の各ノード上のその表内のローカル索引から作られます。複数区画環境で定義されたユニーク索引は、パーティション・キーのスーパーセットでなければならないことに注意してください。

関連概念:

- 「*SQL リファレンス 第 1 巻*」の『索引』
- 149 ページの『索引の使用』
- 150 ページの『CREATE INDEX ステートメントのオプション』
- 154 ページの『ユーザー定義の拡張索引タイプの作成』
- 259 ページの『索引の特権』

関連タスク:

- 11 ページの『索引の作成の並列処理を使用可能にする』
- 148 ページの『索引の作成』
- 206 ページの『既存の表または索引の名前変更』
- 217 ページの『索引、索引の拡張、または索引の指定のドロップ』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『CREATE INDEX ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『CREATE INDEX EXTENSION ステートメント』

索引、索引の拡張、または索引の指定の作成に関する詳細

データベース・マネージャーが保守する索引を処理することも、独自の索引を指定することもできます。

索引の作成

索引 は 1 つ以上のキーのセットであり、それぞれのキーは表の行を指しています。索引では、ポインターを介して直接データへのパスを作成できるので、さらに効率よく表の行にアクセスできます。

手順:

パフォーマンス上のヒント: 以下の一連の作業を実行しようとしている場合には、

1. 表の作成
2. 表のロード
3. 索引の作成 (COLLECT STATISTICS オプションを付けない)
4. RUNSTATS の実行

あるいは、以下の一連の作業を実行しようとしている場合には、

1. 表の作成
2. 表のロード
3. 索引の作成 (COLLECT STATISTICS オプションを付ける)

作業の実行順序を、以下のようにすることを考慮する必要があります。

1. 表を作成する
2. 索引を作成する
3. `statistics yes` オプションを要求して表をロードする

索引作成後も、索引は常に維持されていきます。その結果、アプリケーション・プログラムが、表の中の行をランダムにアクセスおよび処理するためにキー値を使用したときに、そのキー値に基づく索引を使用して、行を直接アクセスすることができます。基本表の中の行の物理的なストレージが順番に並んでいるわけではないので、このことは重要です。

表を作成している時、マルチディメンション・クラスタリング (MDC) 表の作成を選択することができます。このタイプの表を作成することにより、ブロック索引が作成されます。正規の索引は個々の行を指します。ブロック索引はブロックまたはデータのエクステントを指し、ブロック索引は、正規の索引よりもずっと小規模です。ブロック索引は、正規の索引とともに、同じ表スペースに保管されます。

行を挿入すると、クラスタリング索引の定義をしなければ、その行が単に最も便利な保管場所に入れられるだけです。特定の選択条件に一致する表の行を探索しているときで、表に索引がない場合、表全体がスキャンされます。索引を使用すれば、時間のかかる順次探索を実行することなく、データ検索を最適化できます。

索引のデータは、表データと同じ表スペース内か、または索引データが入った別個の表スペース内に保管することができます。索引データの保管に使用する表スペースは、表の作成時に決められます。

コントロール・センターを使用して索引を作成するには、以下のようにします。

1. オブジェクト・ツリーを順に展開し、**「索引 (Indexes)」**フォルダーを表示します。
2. **「索引 (Indexes)」**フォルダーを右クリックして、ポップアップ・メニューから**「作成 (Create)」—>「ウィザードを使用する索引 (Index Using Wizard)」**を選択します。
3. このウィザードのステップに従って、タスクをすべて実行します。

コマンド行を使用して索引を作成するには、以下のように入力します。

```
CREATE INDEX <name> ON <table_name> (<column_name>)
```

関連概念:

- 「*データ移動ユーティリティー ガイドおよびリファレンス*」の『バックアップのパフォーマンスの最適化』
- 149 ページの『索引の使用』
- 150 ページの『CREATE INDEX ステートメントのオプション』
- 259 ページの『索引の特権』

関連タスク:

- 206 ページの『既存の表または索引の名前変更』
- 217 ページの『索引、索引の拡張、または索引の指定のドロップ』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『CREATE INDEX ステートメント』

索引の使用

索引がアプリケーション・プログラムによって直接使用されることはありません。索引を使用するかどうか、および使用できる可能性がある索引はどれかを判断するのは、オプティマイザーの役割です。

表上での最適な索引は、以下の通りです。

- 高速ディスクを使用するもの
- 高クラスター化されたもの
- 狭い範囲だけの列からなるもの
- カーディナリティーが高い列を使用するもの

関連概念:

- 「*管理ガイド: パフォーマンス*」の『索引の計画のヒント』
- 「*管理ガイド: パフォーマンス*」の『索引のパフォーマンスのヒント』
- 「*管理ガイド: パフォーマンス*」の『索引スキャンによるデータ・アクセス』
- 「*管理ガイド: パフォーマンス*」の『標準表における表および索引の管理』
- 「*管理ガイド: パフォーマンス*」の『MDC 表のための表および索引管理』

CREATE INDEX ステートメントのオプション

重複値の許容される索引 (非ユニーク索引) を作成することができます。それによって、主キー以外の列による効率的な検索が可能になり、索引列に重複値を入れることができるようになります。

以下の SQL ステートメントは、EMPLOYEE 表の LASTNAME 列を昇順にソートしたものから、 LNAME という非ユニーク索引を作成します。

```
CREATE INDEX LNAME ON EMPLOYEE (LASTNAME ASC)
```

次の SQL ステートメントは、電話番号列に基づくユニーク索引を作成するものです。

```
CREATE UNIQUE INDEX PH ON EMPLOYEE (PHONENO DESC)
```

ユニーク索引を使う場合は、索引列に重複値を入れることができなくなります。行を更新するかまたは新しい行を挿入する SQL ステートメントの終わりに、制約が施行されます。 1 つ以上の列のセットがすでに重複値をもっている場合には、このタイプの索引は作成できません。

ASC というキーワードは、索引項目を列ごとの昇順にします。また、DESC は、列ごとに降順にします。デフォルトは昇順です。

ユニーク索引を 2 つの列に作成することができます。そのうちの 1 つは組み込み列です。主キーは、組み込み列ではない方の列で定義されます。どちらの列も同じ表での主キーとして、カタログ内に表示されます。通常、1 つの表当たり 1 つの主キーのみです。

INCLUDE 文節では、一連の索引キー列に追加列を付加することを指定します。この文節に組み込まれる列は、固有性を強制するためには使用されません。組み込み列は、索引のみのアクセスを行うことにより、照会のパフォーマンスを向上させることがあります。これらの列は、固有性を強制するために使用される列とは異なっていなければなりません (そうしない場合、エラー・メッセージ SQLSTATE 42711 が出されます)。列の数および長さ属性の合計の限界は、ユニーク・キー内と索引内のすべての列に適用されます。

既存の索引が主キー定義と一致しているかどうか判別するために、チェックが実行されます (索引内の INCLUDE 列はすべて無視されます)。索引の定義が、列の順序または方向 (昇順か降順のいずれか) の指定に関係なく、同じ列のセットを識別する場合は、索引の定義が一致します。一致する索引定義が見つかった場合、システムが求めれば、索引の記述はそれが 1 次索引であることを示すように変更されます。また、それが非ユニーク索引であれば (ユニーク性を確認した後で) ユニークに変更されます。

なぜなら、カタログ内で示された同じ表に複数の主キーを持っている可能性があるからです。

構造化タイプを処理する際、構造化タイプがユーザー定義索引タイプの作成に必要になることがあります。その場合は、索引の保守、索引の検索、および索引活用機能を定義する手段が必要になります。

次の SQL ステートメントは、 EMPLOYEE 表の LASTNAME 列に INDEX1 というクラスタリング索引を作成するものです。

```
CREATE INDEX INDEX1 ON EMPLOYEE (LASTNAME) CLUSTER
```

データベースの内部記憶域を効率的に使用するには、 ALTER TABLE ステートメントの PCTFREE パラメーターでクラスター索引を使用します。このようにすると、新しいデータを正しいページに挿入できます。データが正しいページに挿入されると、クラスタリングの順序が崩れることがありません。通常、表に対する INSERT アクティビティーが多ければ多いほど、クラスタリングを保守するために必要な (その表での) PCTFREE 値は大きくなります。この索引は、データが物理ページに置かれる ORDER BY を決定するので、特定の表について定義できるクラスタリング索引は 1 つのみです。

これらの新しい行の索引キー値が、たとえば常に高いキー値である場合は、表のクラスタリング属性はそのキー値を表の最後に置こうとします。他のページにフリー・スペースがあっても、クラスタリングを保持することについてはほとんど意味がありません。この場合、表を追加モードにすることは、クラスタリング索引および PCTFREE 値を大きくするという方法よりもよい方法です。 ALTER TABLE APPEND ON を出して、表を追加モードにすることができます。

上記の事柄は、UPDATE によって生成される、行のサイズを増加させる新しい「オーバーフロー」行についても当てはまります。

CREATE INDEX ステートメントで ALLOW REVERSE SCANS パラメーターを使用して作成された単一の索引は、順方向または逆方向にスキャンできます。すなわち、そうした索引は、索引が作成された時に定義された方向へのスキャンをサポートし、反対つまり逆方向へのスキャンをサポートします。そのステートメントは、次のようなものになります。

```
CREATE INDEX iname ON tname (cname DESC) ALLOW REVERSE SCANS
```

このケースでは、索引 (iname) は、特定の列 (cname) で、降順の値 (DESC) を基にして形成されます。逆スキャンを許可することにより、列の索引は降順で定義されてはいても、昇順 (逆順) でスキャンを行えます。実際に索引を両方向に使用する時には、それはユーザーによって制御されるのではなく、アクセス・プランを作成および考慮する時のオプティマイザーにより制御されます。

CREATE INDEX ステートメントの MINPCTUSED 文節は、索引リーフ・ページ上で使用される最小スペースの限界値を指定します。この文節が使用されると、この索引用にオンラインの索引デフラグが使用可能になります。それが使用可能になったら、以下の考慮事項を使用して、オンライン索引デフラグを行うかどうかを決定します。すなわち、この索引のリーフ・ページからキーが物理的に除去され、ページ上の使用済みスペースのパーセンテージが指定限界値を下回ったら、近隣の索引リーフ・ページを調べて、 2 つのリーフ・ページ上のキーを単一の索引リーフ・ページにマージできるかどうかを判別します。

たとえば、次の SQL ステートメントは、オンライン索引デフラグが使用可能な索引を作成します。

```
CREATE INDEX LASTN ON EMPLOYEE (LASTNAME) MINPCTUSED 20
```

この索引の索引ページからキーが物理的に除去されるとき、索引ページに残っているキーが索引ページの 20% 以下のスペースを占めていれば、この索引ページのキーと近隣の索引ページのキーをマージして索引ページを削除しようとします。結合したキーが単一ページにすべて収まる場合、このマージは実行され、索引ページの 1 つが削除されます。

CREATE INDEX ステートメントを使用すると、基礎表および任意の以前の既存の索引に読み取りアクセスおよび書き込みアクセスを行えるようになるのと同時に、索引を作成できるようになります。索引の作成中に表へのアクセスを制限するために、 LOCK TABLE ステートメントを使用して索引を作成する前に表をロックします。新規索引は、基礎表をスキャンすることにより作成されます。索引作成中に表に対してなされたすべての変更は、ログに記録されます。これらの変更は、新規索引が作成されたとき、その索引に適用されます。索引を作成する際に、ログに記録された変更をより速く適用できるよう、ユーティリティー・ヒープからオンデマンドで割り振られるメモリー・バッファー・スペースに、変更点の別のコピーが保持されます。これによって、索引の作成は、まずメモリーから変更を直接読み取ることによって処理し、必要な場合には、もっと後でログ全体を読み取ることによって実行できます。すべての変更がその索引に適用されると、新規索引が可視になっている間、その表は静止します。

ユニーク索引を作成する時には、表の中に複写キーがないように、さらに索引作成中に並行挿入により複写キーを取り入れることがないようにしてください。索引作成時には、据え置きのユニーク・スキームを使用して複写キーを検出するため、索引作成の最後まで複写キーは検出されません。その時点で、複写キーのために索引作成は失敗します。

CREATE INDEX ステートメント PCTFREE 文節は、索引が作成されるときにフリー・スペースとして残す各索引ページのパーセンテージを指定します。索引ページに多くのフリー・スペースを残すと、分割されるページが少なくなります。このようにすると、順次索引ページを回復するために表を再編成する必要性が少なくなりますが、この必要性はプリフェッチを増大させるものです。プリフェッチは、パフォーマンスを向上させる 1 つの重要なコンポーネントです。また、キー値が常に高い場合は、 CREATE INDEX ステートメントの PCTFREE 文節の値を低くすることを考慮したいでしょう。そうすることにより、各索引ページ上で予約される浪費スペースは限られたものになります。

LEVEL2 PCTFREE 文節は、索引の 2 次レベルの各ページに、指定したパーセンテージのフリー・スペースを保持するよう、システムに指示します。フリー・スペースのパーセンテージは、索引を作成する際に、将来の挿入や更新を考慮して指定しておきます。 2 次レベルとは、リーフ・レベルのすぐ上のレベルを言います。デフォルトでは、すべての非リーフ・ページで、最低 10 % と PCTFREE の値が保持されます。 LEVEL2 PCTFREE パラメーターでは、デフォルトを上書きできます。CREATE INDEX ステートメントの中で LEVEL2 PCTFREE 整数オプションを使用すると、レベル 2 の中間ページに、整数で指定されたパーセントのフリー・スペースが残されます。レベル 3 以上の中間ページでは、最低 10 % と、指定された整数のパーセントのフリー・スペースが残されます。 2 次レベルに残しておくスペースが大きければ大きいほど、索引の 2 次レベルで行われるページ分割の数は少なくなります。

PAGE SPLIT SYMMETRIC、PAGE SPLIT HIGH、および PAGE SPLIT LOW といった文節を使用すると、索引への挿入を行う際の、ページ分割の振る舞いを選択できます。

PAGE SPLIT SYMMETRIC 文節は、デフォルトのページ分割の振る舞いで、索引ページの中間を大まかに分割します。このデフォルトの振る舞いは、索引への追加がランダムな場合や、 PAGE SPLIT HIGH 文節や PAGE SPLIT LOW 文節で対象にされているいずれのパターンにも従わない場合に、最適です。

PAGE SPLIT HIGH の振る舞いは、索引内に増加し続ける範囲がある場合に便利です。このように増加し続ける範囲は、次のような場合に発生します。

- 複数のキー・パーツを持つ索引があり、たくさんの値 (複数の索引ページ分) がある場合で、最後のキー・パーツ以外のすべてのキー・パーツが同じ値になっているとき
- 表への挿入が、すべて、最後のキー・パーツを除くすべての既存のキーと同じ値を持つ新しい値で構成されているとき
- 挿入された値の最後のキー・パーツが、既存のキーの最後のキー・パーツより大きいとき

たとえば、索引内に次のようなキー値があるとします。

```
(1,1),(1,2),(1,3), ... (1,n),
(2,1),(2,2),(2,3), ... (2,n),
...
(m,1),(m,2),(m,3), ...(m,n)
```

そして、次に挿入されるキーには、1 <= x <= m および y > n の値 (x,y) が含まれます。このようなパターンでキーの挿入が続いていく場合は、PAGE SPLIT HIGH 文節を使用して、ページ分割の結果として多くのページが 50 % 空きの状態にならないようにできます。

同様に、PAGE SPLIT LOW は、索引内に減少し続ける範囲があるとき、ページの空きが 50 % になるのを防ぐために使用できます。

注: 1 次キーやユニークなキーを追加する場合、および基礎になる索引に SPLIT HIGH、SPLIT LOW、PCTFREE、LEVEL2 PCTFREE、 MINPCTUSED、CLUSTER、または ALLOW REVERSE SCANS を使用させる場合は、まず、希望するキーやパラメーターを指定した索引を作成する必要があります。次いで、ALTER TABLE ステートメントを使用して、1 次キーやユニーク・キーを追加します。 ALTER TABLE ステートメントは、既に作成してある索引を選出し、再利用します。

索引作成の一部として索引統計を収集することができます。 CREATE INDEX ステートメントを使用する時、キー値統計および物理統計を入手して使用することができます。索引統計を CREATE INDEX ステートメントの一部として収集することにより、 CREATE INDEX ステートメントの完了直後に RUNSTATS ユーティリティーを実行する必要はありません。

たとえば、次の SQL ステートメントは、基本索引統計を索引作成の一部として収集します。

```
CREATE INDEX IDX1 ON TABL1 (COL1) COLLECT STATISTICS
```

複製サマリー表がある場合は、その基本表 (複数も可) にはユニーク索引がなければならず、複製サマリー表を定義する照会で索引キー列が使用される必要があります。

パーティション内並列処理の場合、索引作成のパフォーマンスは、索引作成の間に実行されるデータのスキャンとソートに複数のプロセッサーを使用することによって向上します。複数プロセッサーの使用は、 *intra_parallel* を YES(1) または ANY(-1) に設定することによって可能になります。索引作成の間に使用されるプロセッサーの数はシステムによって決定され、構成パラメーターの *dft_degree* または *max_querydegree*、アプリケーションの実行時の程度、または SQL ステートメントのコンパイルの程度によって影響を受けることはありません。

複数パーティション・データベースでは、ユニーク索引をパーティション・キーのスーパーセットとして定義しなければなりません。

関連概念:

- 「*管理ガイド: パフォーマンス*」の『索引のパフォーマンスのヒント』
- 「*管理ガイド: パフォーマンス*」の『索引の再編成』
- 「*管理ガイド: パフォーマンス*」の『標準表における表および索引の管理』
- 「*管理ガイド: パフォーマンス*」の『オンライン索引のデフラグ』
- 「*管理ガイド: パフォーマンス*」の『MDC 表のための表および索引管理』

関連タスク:

- 202 ページの『表属性の変更』

関連資料:

- 「*管理ガイド: パフォーマンス*」の『max_querydegree - 「照会の最大並列処理の度合い」構成パラメーター』
- 「*管理ガイド: パフォーマンス*」の『intra_parallel - 「パーティション内並列処理の使用可能化」構成パラメーター』
- 「*管理ガイド: パフォーマンス*」の『dft_degree - 「デフォルトの並列処理の度合い」構成パラメーター』
- 「*SQL リファレンス 第 2 巻*」の『CREATE INDEX ステートメント』

ユーザー定義の拡張索引タイプの作成

DB2® Universal Database では、ユーザー定義の索引タイプをサポートするため、索引がどのように機能するかを制御する主なコンポーネントの論理を、独自に作成して適用できます。この種の置換可能なコンポーネントには、以下のものが含まれます。

- 索引の保守。これは、索引列の内容を索引キーにマップすることを可能にします。この種のマッピングは、ユーザー定義のマッピング関数を使用して行われます。 1 つの拡張索引に関与できる構造型列は 1 つだけです。拡張索引は、通常の索引とは異なり、 1 行あたり複数の索引項目を持つことができます。行ごとに複数の索引項目を使用することにより、たとえば、テキスト文書を 1 つのオブジェクトとして格納し、文書中の各キーワードを別々の索引項目とすることができます。

- 索引活用。アプリケーション設計者は、これを使用すると、フィルター条件 (範囲述部) をユーザー定義関数 (UDF) と関連付けることができます。そうしない場合、UDF はオプティマイザーから隠されます。これにより、DB2 は行ごとに別々に UDF 呼び出しを行わずに済むので、クライアントとサーバーの間でコンテキストを切り替える必要がなくなり、パフォーマンスが大幅に向上します。

注: ユーザー定義関数の定義は決定論的でなければならず、外部アクションを許可して、オプティマイザーから活用できるようにしてはなりません。

オプションでデータ・フィルター関数も指定できます。オプティマイザーは、取り出されたタプルに対してフィルターを使用してから、ユーザー定義関数を評価します。

構造型または特殊タイプの列に限り、索引の拡張を使用して、これらのオブジェクトに対してユーザー定義の拡張型タイプを作成できます。ユーザー定義の拡張索引タイプには以下のことは行えません。

- クラスター索引と共に定義すること。
- INCLUDE 列を持つこと。

関連概念:

- 155 ページの『索引の保守に関する詳細』
- 156 ページの『索引の検索に関する詳細』
- 157 ページの『索引活用に関する詳細』
- 158 ページの『索引の拡張を定義するシナリオ』

ユーザー定義の拡張索引タイプの作成に関する詳細

このセクションでは、独自の拡張索引タイプを作成する際に必要な、さまざまな局面を説明します。

索引の保守に関する詳細

CREATE INDEX EXTENSION ステートメントを使用して、索引の操作を制御する 2 つのコンポーネントを定義できます。

索引の保守とは、索引列の内容 (またはソース・キー) をターゲットの索引キーに変換するプロセスのことです。変換プロセスは、データベースに事前定義された表関数を使用して定義されます。

FROM SOURCE KEY 文節は、この索引の拡張でサポートされるソース・キー列のタイプを、構造型データ・タイプまたは特殊タイプとして指定します。パラメーター名とデータ・タイプを 1 つずつ指定して、それらをソース・キー列と関連付けます。

GENERATE KEY USING 文節は、索引キーの生成に使用されるユーザー定義表関数を指定します。この関数からの出力の指定は、 TARGET KEY 文節の指定の中で行わなければなりません。 この関数からの出力を、 FILTER USING 文節で指定される索引フィルター関数の入力として使用することもできます。

関連概念:

- 154 ページの『ユーザー定義の拡張索引タイプの作成』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『CREATE INDEX EXTENSION ステートメント』

索引の検索に関する詳細

索引の検索は、検索引き数を検索範囲にマップします。

CREATE INDEX EXTENSION ステートメントの WITH TARGET KEY 文節は、ターゲット・キー・パラメーターを指定します。このパラメーターは、GENERATE KEY USING 文節に指定されるユーザー定義表関数の出力です。パラメーター名とデータ・タイプを 1 つずつ指定して、それらをターゲット・キー列と関連付けます。このパラメーターは、 GENERATE KEY USING 文節のユーザー定義表関数の RETURNS 表の列に対応します。

SEARCH METHODS 文節は、索引の検索の方式を 1 つまたは複数定義します。検索の方式はそれぞれ、方式名、検索引き数、範囲作成関数、およびオプションの索引フィルター関数から成ります。検索の方式はそれぞれ、ユーザー定義表関数が、基礎となるユーザー定義索引の索引検索範囲をどのように作成するかを定義します。さらに、検索の方式はそれぞれ、特定の検索範囲にある索引項目が、単一値を戻すためにユーザー定義のスカラー関数によってさらに限定される方法を定義します。

- WHEN 文節は、ラベルを検索の方式と関連付けます。ラベルとは、SQL ID の一種で、索引活用規則 (ユーザー定義関数の PREDICATES 文節にある) で指定された方式名に関連しています。範囲関数 (索引フィルター関数を含む場合も含まない場合も) の引き数として使用するため、 1 つまたは複数のパラメーター名とデータ・タイプを指定します。 WHEN 文節は、 CREATE FUNCTION ステートメントの PREDICATES 文節と着信する照会とが合致した場合に、オプティマイザーによって取られるアクションを指定します。
- RANGE THROUGH 文節は、索引キーの範囲を作成するユーザー定義の外部表関数を指定します。この文節を使用すると、索引キーがキー範囲外にある場合に、関連する UDF をオプティマイザーが呼び出さないようにできます。
- FILTER USING 文節は、範囲作成関数から戻される索引項目をフィルターに掛ける、ユーザー定義の外部表関数または case 式を指定します (オプション)。索引フィルター関数または case 式から値 1 が戻された場合は、索引項目に対応する行が表から取り出されます。 1 以外の値が戻された場合、索引項目は廃棄されます。この機能が有用なのは、2 次フィルターの費用が元の方式の評価の費用に比べて低く、かつ 2 次フィルターの選択率が比較的低い場合です。

関連概念:

- 154 ページの『ユーザー定義の拡張索引タイプの作成』
- 155 ページの『索引の保守に関する詳細』
- 157 ページの『索引活用に関する詳細』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『CREATE INDEX EXTENSION ステートメント』

索引活用に関する詳細

索引活用は、検索の方式の評価時に生じます。

CREATE FUNCTION (外部スカラー) ステートメントは、索引の拡張のために定義された検索の方式と共に使用される、ユーザー定義の述部を作成します。

PREDICATES 文節は、この関数が使用されている述部、つまり索引の拡張を活用する可能性のある (および検索条件のためにオプションの SELECTIVITY 文節を使用する可能性のある) 述部を識別します。 PREDICATES 文節を指定する場合は、NO EXTERNAL ACTION を指定して、関数を DETERMINISTIC として定義しなければなりません。

- WHEN 文節の後ろには、述部の中で比較演算子 (=、>、 < など) および定数または式 (EXPRESSION AS 文節を使用) と共に定義される関数の、特定の使用法が続きます。述部でこの関数が、同じ比較演算子、および所定の定数または式と共に使用される場合、フィルター操作や索引活用を使用できます。定数は、主に結果タイプ 1 または 0 のブール式を処理するために使用します。それ以外の場合はいずれも、EXPRESSION AS 文節を選択する方が賢明です。
- FILTER USING 文節は、結果表に追加のフィルター操作を実行するために使用できるフィルター関数を識別します。これは、定義される関数 (述部で使用される関数) に代わる、より高速な関数です。適格かどうかを判断するためにユーザー定義の述部を実行しなければならない行数を削減します。索引によって作成された結果が、ユーザー定義の述部が予期する結果と近い場合、このフィルター関数の適用は余分なものとなる可能性があります。
- オプションで、索引を活用するために、索引の拡張の検索の方式ごとに一連の規則を定義することができます。また、索引の拡張に検索の方式を定義し、検索のターゲット、検索引き数、および索引の検索を実行する際の使用法を記述することもできます。
  - SEARCH BY INDEX EXTENSION 文節は、索引の拡張を識別します。
  - オプションの EXACT 文節は、述部の評価の点で、索引の検索が厳密に行われることを示します。この文節は、索引の検索の後、元のユーザー提供の述部関数またはフィルター関数を適用しないよう、データベースに指示します。索引の検索を使用しない場合は、元の述部とフィルター関数を適用する必要があります。 EXACT 文節を使用しないと、索引の検索の後で、元のユーザー提供の述部が適用されます。 EXACT 述部は、索引の検索により戻される結果が、述部と同じである場合に効果的です。この述部により、照会の実行時に、索引検索から得られる結果にユーザー定義の述部が適用されないようにすることができます。索引が述部に近似しているに過ぎないと予想される場合は、 EXACT 文節を指定しないでください、
  - WHEN KEY 文節は、検索のターゲットを定義します。キー当たり 1 つだけ検索ターゲットを指定できます。 WHEN KEY 文節の後に指定する値は、定義しようとしている関数のパラメーター名を識別します。指定されたパラメーターの値が、指定された索引の拡張に基づく索引に含まれる列である場合、この文節は真として評価されます。
  - USE 文節は、検索引き数を定義します。検索引き数は、索引の拡張で定義されている方式のうちどれを使用するかを識別します。ここに指定する方式名は、索引の拡張に定義されている方式と一致していなければなりません。 1 つま

たは複数のパラメーター値により、定義する関数のパラメーター名を識別します。この値は、検索ターゲットに指定されているパラメーター名とは別の値でなければなりません。パラメーター値の数とその個々のデータ・タイプは、索引の拡張中に定義されている方式のパラメーターと一致していなければなりません。組み込みデータ・タイプと特殊データ・タイプの場合、一致は厳密でなければならず、同じ構造型に属していなければなりません。

関連概念:

- 154 ページの『ユーザー定義の拡張索引タイプの作成』
- 155 ページの『索引の保守に関する詳細』
- 156 ページの『索引の検索に関する詳細』
- 158 ページの『索引の拡張を定義するシナリオ』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『CREATE FUNCTION (外部スカラー) ステートメント』

索引の拡張を定義するシナリオ

索引の拡張の定義のシナリオを以下に示します。

1. 構造型 (形状) を定義します。 CREATE TYPE ステートメントを使用してタイプ階層を定義します。形状はスーパータイプで、形状なし、点、線、および多角形はサブタイプです。これらの構造型は、空間のエンティティーをモデル化します。たとえば、店の場所は点、河川の流域は線、営業上の地区の境界は多角形になります。最低限の境界を持つ長方形 (minimum bounded rectangle: mbr) は 1 つの属性です。 gtype 属性は、関連したエンティティーが点、線、または多角形のいずれであるかを識別します。地理的な境界は、 numpart、numpoint、および geometry 属性によってモデル化されます。これら以外の属性は、このシナリオとは関係ないので無視します。
2. 索引の拡張を作成します。
   - CREATE FUNCTION ステートメントを使用して、キー変換 (gridentry)、範囲の作成 (gridrange)、および索引のフィルター操作 (checkduplicate と mbroverlap) に使用する関数を作成します。
   - CREATE INDEX EXTENSION ステートメントを使用して、索引のコンポーネントとして必要なもののうち、残りのものを作成します。
3. 索引の保守コンポーネントに対応するキー変換を作成します。

   ```
   CREATE INDEX EXTENSION iename (parm_name datatype, ...)
      FROM SOURCE KEY (parm_name datatype)
      GENERATE KEY USING table_function_invocation
      ...
   ```

   FROM SOURCE KEY 文節は、キー変換のパラメーターとデータ・タイプを識別します。 GENERATE KEY USING 文節は、自分が生成した値とソース・キーとをマップする関数を識別します。
4. 索引の検索コンポーネントに対応する、範囲作成関数と索引フィルター関数を定義します。

```
CREATE INDEX EXTENSION iename (parm_name datatype, ...)
   ...
   WITH TARGET KEY
      WHEN method_name (parm_name datatype, ...)
      RANGE THROUGH range_producing_function_invocation
      FILTER USING index_filtering_function_invocation
```

WITH TARGET KEY 文節は、検索の方式の定義を識別します。 WHEN 文節は、方式名を識別します。 RANGE THROUGH 文節は、使用される索引の有効範囲を制限するのに使用される関数を識別します。 FILTER USING 文節は、結果の索引値から不要な項目を除去するのに使用される関数を識別します。

注: FILTER USING 文節は、索引フィルター関数の代わりに case 式を識別することもできます。

5. 索引の拡張を活用するための述部を定義します。

```
CREATE FUNCTION within (x shape, y shape)
   RETURNS INTEGER
   ...
   PREDICATES
      WHEN = 1
         FILTER USING mbrWithin (x..mbr..xmin, ...)
         SEARCH BY INDEX EXTENSION grid_extension
         WHEN KEY (parm_name) USE method_name(parm_name)
```

PREDICATES 文節の後ろには、1 つまたは複数の述部が続きます。それぞれの述部は、WHEN 文節で始まります。 WHEN 文節は、比較演算子とその後に続く定数または EXPRESSION AS 文節で始まり、その後に述部の指定が続きます。 FILTER USING 文節は、結果表に追加のフィルター操作を実行するために使用できるフィルター関数を識別します。これは、定義される関数 (述部で使用される関数) に代わる、より費用の低い関数です。適格な行を判別するためにユーザー定義の述部を実行しなければならない行数を削減します。 SEARCH BY INDEX EXTENSION 文節は、索引活用を実行する場所を指定します。索引活用は、索引を活用するために使用できる索引の拡張の検索の方式を使用して、一連の規則を定義します。 WHEN KEY 文節は、活用の規則を指定します。活用の規則として、検索ターゲット、検索引き数、および検索の方式を使用して索引検索を実行する際の使用法を記述できます。

6. フィルター関数を定義します。

```
CREATE FUNCTION mbrWithin (...)
```

ここで定義されている関数は、索引の拡張の述部で使用するために作成されます。

照会のパフォーマンスを向上するために作成した索引が、照会オプティマイザーにより正常に活用されるようにするため、関数呼び出しに SELECTIVITY オプションを使用できます。述部によって戻される行のパーセンテージを指定したい場合は、関数呼び出しに SELECTIVITY オプションを使用して、 DB2® オプティマイザーが有効なアクセス・パスを選択するようにすることができます。

以下の例で、 within ユーザー定義関数は中心と半径を (中心は 1 つ目のパラメーターに基づいて、半径は 2 つ目のパラメーターに基づいて) 計算し、該当する選択率でステートメントのストリングを構築します。

```
SELECT * FROM customer
   WHERE within(loc, circle(100, 100, 10)) = 1 SELECTIVITY .05
```

この例に示されている述部 (SELECTIVITY .05) は、 customer 表の行のうち 95% をフィルターに掛けて除きます。

関連概念:

- 154 ページの『ユーザー定義の拡張索引タイプの作成』
- 155 ページの『索引の保守に関する詳細』
- 156 ページの『索引の検索に関する詳細』
- 157 ページの『索引活用に関する詳細』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『CREATE INDEX EXTENSION ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『CREATE FUNCTION (外部スカラー) ステートメント』

コマンド行プロセッサーからの構成アドバイザーの起動

前提条件:

データベースが作成されている必要があります。

手順:

データベースを作成した後、AUTOCONFIGURE コマンドを使用して構成アドバイザーを起動できます。この方式は、データベースの作成時に AUTOCONFIGURE オプションを選択している場合でも使用できます。

AUTOCONFIGURE で使用可能なオプションを使用して、いくつかの構成パラメーターに値を定義したり、これらのパラメーターのアプリケーションの有効範囲を決定することができます。有効範囲は、NONE (どの値も適用されないという意味)、DB ONLY (データベース構成およびバッファー・プール値のみが適用されるという意味)、または DB AND DBM (すべてのパラメーターおよびその値が適用されるという意味) になります。

関連概念:

- 「*管理ガイド: パフォーマンス*」の『構成パラメーター』

関連資料:

- 「コマンド・リファレンス」の『AUTOCONFIGURE コマンド』

第 5 章 データベースの変更

この章では、データベースを変更する前に考慮しなければならない点、およびデータベース・オブジェクトの変更方法とドロップ方法について説明します。

インスタンスの変更

データベース設計が実現した後に、データベース設計の変更が必要になる場合があります。以前の設計での主要な設計上の論点を再考慮する必要が生じます。

データベース全体に影響を与える変更を行う前に、すべての論理的および物理的な設計の決定事項について見直す必要があります。たとえば、1 つの表スペースを変更する場合、SMS または DMS のストレージ・タイプの使用に関して、設計上の決定事項の見直しを行う必要があります。

DB2 Universal Database™ (DB2 UDB) 製品のライセンス管理の一環として、ライセンスの数を増やす必要があるかもしれません。コントロール・センターの中のライセンス・センターを使用すれば、インストールした製品の使用状況を調べ、その使用状況に基づいてライセンスの数を増やすことができます。

以下の点に特別な注意を払う必要があります。

- 『インスタンスの変更 (UNIX のみ)』
- 166 ページの『ノードおよびデータベース構成ファイルの変更』

インスタンスの変更 (UNIX のみ)

インスタンスは、後ほど製品をインストールしたり削除したりしても、できるだけ影響を受けないように設計されています。

既存のインスタンスはたいてい、インストールまたは削除する製品の機能を自動的に継承するか、アクセスを失うかのどちらかです。しかし、特定の実行可能コードやコンポーネントがインストールまたは削除された場合、既存のインスタンスは自動的に新しいシステムの構成パラメーターを継承したり、すべての追加機能へのアクセスを取得したりするわけではありません。そのインスタンスは更新しなければなりません。

プログラム一時修正 (PTF) またはパッチを使って DB2® Universal Database (DB2 UDB) を更新する場合は、 **db2iupdt** コマンドを使って既存のすべての DB2 UDB インスタンスを更新する必要があります。

インスタンスを変更または削除する前には、インスタンス内にあるインスタンス・サーバーとデータベース・パーティション・サーバーを理解しておく必要があります。

関連概念:

- 17 ページの『インスタンスの作成』

関連タスク:

- 162 ページの『UNIX でのインスタンス構成の更新』
- 165 ページの『インスタンスの除去』

関連資料:

- 「コマンド・リファレンス」の『db2iupdt - インスタンスの更新コマンド』

インスタンスの変更に関する詳細

インスタンスを変更する前に、すべての既存のインスタンスをリストする必要があります。

インスタンスのリスト

手順:

コントロール・センターを使用して、システムで使用可能なインスタンスをすべてリストするには、次のようにします。

1. オブジェクト・ツリーを順に展開し、**「インスタンス (Instances)」**フォルダーを表示します。
2. インスタンスのフォルダーを右クリックして、ポップアップ・メニューから**「追加 (Add)」**を選択します。
3. 「インスタンスの追加 (Add Instance)」ウィンドウで、**「最新表示 (Refresh)」**をクリックします。
4. ドロップダウン矢印をクリックして、データベース・インスタンスのリストを表示します。
5. **「キャンセル (Cancel)」**をクリックして、ウィンドウを終了します。

コマンド行を使用して、システムで使用可能なインスタンスをすべてリストするには、次のようにします。

```
db2ilist
```

(サポートされている Windows プラットフォーム上で) 現行セッションに適用されるインスタンスを判別するには、次のコマンドを使用します。

```
set db2instance
```

UNIX でのインスタンス構成の更新

db2iupdt コマンドを実行すると、以下の事柄を実行して指定インスタンスが更新されます。

- インスタンス所有者のホーム・ディレクトリー下の `sqllib` サブディレクトリーにあるファイルを置き換える。
- ノード・タイプが変更されていれば、新しいデータベース・マネージャー構成ファイルが作成される。この作成は、既存のデータベース・マネージャー構成ファイルから関連する値を、新しいノード・タイプ用のデフォルトのデータベース・マネージャー構成ファイルとマージして行います。新しいデータベース・マネー

ジャー構成ファイルが作成されると、古いファイルは、インスタンス所有者のホーム・ディレクトリーにある sqllib サブディレクトリーの backup サブディレクトリーにバックアップされます。

手順:

db2iupdt コマンドは、 AIX では /usr/opt/db2_08_01/instance/ ディレクトリーにあります。 **db2iupdt** コマンドは、HP-UX、Solaris オペレーティング環境、または Linux では /opt/IBM/db2/V8.1/instance/ ディレクトリーにあります。

コマンドは、次のように使用します。

```
db2iupdt InstName
```

InstName はインスタンス所有者のログイン名です。

このコマンドに関連して、他にも次の任意指定パラメーターがあります。

- -h または -?

  このコマンドのヘルプ・メニューを表示します。

- -d

  問題判別中に使用するデバッグ・モードを設定します。

- -a AuthType

  インスタンスの認証タイプを指定します。有効な認証タイプは、SERVER、SERVER_ENCRYPT、または CLIENT です。指定しない場合、DB2 サーバーがインストールされていれば、デフォルトは SERVER に設定されます。それ以外の場合は、CLIENT に設定されます。インスタンスの認証タイプは、インスタンスが所有するすべてのデータベースに適用されます。

- -e

  既存の各インスタンスを更新できます。存在するものは **db2ilist** を使用して表示できます。

- -u Fenced ID

  fenced ユーザー定義関数 (UDF) とストアード・プロシージャーを実行するユーザーの名前を指定します。 DB2 クライアントまたは DB2 Software Developer's Kit をインストールする場合、これは必須ではありません。 他の DB2 製品の場合、これは必要パラメーターです。

  注: Fenced ID には、″root″ または ″bin″ は使えません。

- -k

  このパラメーターは、現在のインスタンス・タイプを保存します。このパラメーターを指定しない場合、現行インスタンスは次の順序で、使用可能な上位のインスタンス・タイプにアップグレードされます。

  – ローカルおよびリモート・クライアントを持つパーティション・データベース・サーバー (DB2 Enterprise - Extended Edition のデフォルト・インスタンス・タイプ)

– ローカルおよびリモート・クライアントを持つデータベース・サーバー (DB2 Universal Database Enterprise Server Edition のデフォルト・インスタンス・タイプ)
– クライアント (DB2 クライアントのデフォルト・インスタンス・タイプ)

次に例を示します。

- インスタンスの作成後に DB2 Universal Database Workgroup Server Edition または DB2 Universal Database Enterprise Server Edition をインストールした場合は、次のコマンドを入力してインスタンスを更新してください。

```
db2iupdt -u db2fenc1 db2inst1
```

- DB2 Connect Enterprise Edition をインストールした場合は、インスタンス名を fenced ID としても使用できます。

```
db2iupdt -u db2inst1 db2inst1
```

- クライアント・インスタンスを更新するには、次のコマンドを使用することができます。

```
db2iupdt db2inst1
```

関連タスク:

- 165 ページの『インスタンスの除去』

関連資料:

- 「コマンド・リファレンス」の『db2ilist - インスタンスのリスト・コマンド』
- 「コマンド・リファレンス」の『db2iupdt - インスタンスの更新コマンド』

Windows でのインスタンス構成の更新

db2iupdt コマンドを実行すると、以下の事柄を実行して指定インスタンスが更新されます。

- インスタンス所有者のホーム・ディレクトリー下の sqllib サブディレクトリーにあるファイルを置き換える。
- ノード・タイプが変更されていれば、新しいデータベース・マネージャー構成ファイルが作成される。この作成は、既存のデータベース・マネージャー構成ファイルから関連する値を、新しいノード・タイプ用のデフォルトのデータベース・マネージャー構成ファイルとマージして行います。新しいデータベース・マネージャー構成ファイルが作成されると、古いファイルは、インスタンス所有者のホーム・ディレクトリーにある sqllib サブディレクトリーの backup サブディレクトリーにバックアップされます。

手順:

db2iupdt コマンドは、 ¥sqllib¥bin ディレクトリーにあります。

コマンドは、次のように使用します。

```
db2iupdt InstName
```

InstName はインスタンス所有者のログイン名です。

このコマンドに関連して、他にも次の任意指定パラメーターがあります。

- /h: hostname

現行マシンに 1 つ以上の TCP/IP ホスト名がある場合に、デフォルトの TCP/IP ホスト名をオーバーライドします。

- /p: instance profile path

  更新されたインスタンスに対して新規のインスタンス・プロファイル・パスを指定します。

- /r: baseport,endport

  複数パーティションで稼働している時に、パーティション・データベース・インスタンスにより使用される TCP/IP ポートの範囲を指定します。

- /u:username,password

  DB2 サービスのアカウント名とパスワードを指定します。

関連タスク:

- 27 ページの『インスタンスのリスト』
- 162 ページの『UNIX でのインスタンス構成の更新』
- 165 ページの『インスタンスの除去』

インスタンスの除去

手順:

コントロール・センターを使用してインスタンスを除去するには、以下のようにします。

1. オブジェクト・ツリーを順に展開し、除去したいインスタンスを表示します。
2. インスタンス名を右クリックして、ポップアップ・メニューから**「除去 (Remove)」**を選択します。
3. **「確認 (Confirmation)」**ボックスにチェックを付け、**「OK」**をクリックします。

コマンド行を使用してインスタンスを除去するには、以下のように入力します。

```
db2idrop <instance_name>
```

コマンド行を使用してインスタンスを除去する場合の、準備事項と詳細は以下のとおりです。

1. インスタンスを現在使用しているアプリケーションをすべて停止します。
2. それぞれの DB2 コマンド・ウィンドウで **db2 terminate** コマンドを実行して、コマンド行プロセッサーを停止します。
3. **db2stop** コマンドを実行してインスタンスを停止します。
4. DB2INSTPROF レジストリー変数で指示されたインスタンス・ディレクトリーをバックアップします。

   UNIX オペレーティング・システムでは、 INSTHOME/sqllib ディレクトリー (ただし、INSTHOME は、インスタンス所有者のホーム・ディレクトリー) にファイルをバックアップすることを考慮します。たとえば、データベース・マネージ

ャー構成ファイル db2systm、 db2nodes.cfg ファイル、ユーザー定義関数 (UDF)、または fenced ストアード・プロシージャー・アプリケーションを保管したいと思うかもしれません。

5. (UNIX オペレーティング・システム上のみ) インスタンス所有者としてログオフします。
6. (UNIX オペレーティング・システム上のみ) root 権限を持つユーザーとしてログインします。
7. 次のように **db2idrop** コマンドを発行します。

   ```
   db2idrop InstName
   ```

   ここで、InstName は、ドロップされるインスタンスの名前です。

   このコマンドを使うと、インスタンスのリストからインスタンス項目が除去され、インスタンス・ディレクトリーが除去されます。
8. (UNIX オペレーティング・システム上のみ) root 権限を持つユーザーとして、このインスタンス所有者のユーザー ID およびグループ (そのインスタンスだけに使用している場合) を除去することもできます。インスタンスの再作成を計画している場合は、上記の除去は行わないでください。

   インスタンス所有者およびインスタンス所有者グループは他の目的で使用することもあるので、このステップはオプションです。

db2idrop コマンドを使うと、インスタンスのリストからインスタンス項目が除去され、インスタンス所有者のホーム・ディレクトリーにある sqllib サブディレクトリーが除去されます。

注: UNIX オペレーティング・システムでは、 db2idrop コマンドを使用してインスタンスをドロップしようとする時、 sqllib サブディレクトリーはドロップできないという内容のメッセージが生成され、 adm サブディレクトリー内に .nfs 拡張子が付いた幾つものファイルが生成されます。 adm サブディレクトリーは NFS でマウントされたシステムで、ファイルはサーバー上で制御されます。ディレクトリーがマウントされているところにあるファイル・サーバーから *.nfs ファイルを削除する必要があります。その後、sqllib サブディレクトリーを除去できます。

関連資料:

- 「コマンド・リファレンス」の『db2stop - DB2 の停止コマンド』
- 「コマンド・リファレンス」の『TERMINATE コマンド』
- 「コマンド・リファレンス」の『STOP DATABASE MANAGER コマンド』
- 「コマンド・リファレンス」の『db2idrop - インスタンスの除去コマンド』
- 「コマンド・リファレンス」の『db2ilist - インスタンスのリスト・コマンド』

ノードおよびデータベース構成ファイルの変更

データベース構成ファイルを更新するには、コントロール・センター内の構成アドバイザーを使用するか、 *db2 autoconfigure* に適切なオプションを付けて実行します。構成アドバイザーは、どの構成パラメーターを修正したらよいかを提案し、そ

れらの推奨値を示すことによって、インスタンスごとの単一データベースのパフォーマンスのチューニングとメモリー所要量のバランスをとるよう支援します。

注: パラメーターを修正しても、以下の時点まで値は更新されません。

- データベース・パラメーターの場合、すべてのアプリケーションが切断された後で、そのデータベースに対する最初の新しい接続が行われるまで。
- データベース・マネージャー・パラメーターの場合、次回、そのインスタンスを停止して開始するまで。

ほとんどのケースで、構成アドバイザーによって推奨された値は、デフォルト値よりもパフォーマンスが良くなります。これは、ユーザーのワークロードとユーザー自身の固有のサーバーについての情報に基づいた値であるためです。ただし、この値は、指定されたデータベース・システムのパフォーマンスを改善するよう設計されたものであり、必ずしも最適なものではありません。これらの値は、最適なパフォーマンスを獲得するために、さらに調整を行うための出発点と考えてください。

前提条件:

いずれかのデータベース・パーティションを変更 (パーティションの追加・削除、または既存のパーティションの移動) する計画を立てている場合には、ノード構成ファイルを更新する必要があります。

データベースに対する変更を計画している場合、構成パラメーターの値を見直す必要があります。一部の値は、その使われ方に応じて、データベースに対して行われる現在進行中の変更の一環として、定期的に調整できます。

手順:

コントロール・センターを使用してデータベース構成を更新するには、以下のようにします。

1. オブジェクト・ツリーを順に展開し、**「データベース (Databases)」**フォルダーを表示します。
2. 変更するデータベースのインスタンスを右クリックし、**「構成アドバイザー」**をクリックします。
3. 個々のページをクリックし、必要に応じて情報に変更を加えます。
4. **「結果 (Results)」**ページをクリックし、その構成パラメーターに推奨されている変更をすべて確認します。
5. 更新内容を適用または保管する準備ができたら、**「完了 (Finish)」**をクリックします。

コマンド行から構成アドバイザーを使用する場合は、AUTOCONFIGURE コマンドを使用します。

コマンド行を使用してデータベース・マネージャー構成の個々のパラメーターを更新するには、以下のようにします。

```
UPDATE DBM CFG FOR <database_alias>
   USING <config_keyword>=<value>
```

1 つのコマンドで 1 つまたは複数の <config_keyword>=<value> の組み合わせを更新できます。データベース・マネージャー構成ファイルに対する変更内容は、その

ほとんどが、メモリーへのロード後にはじめて有効になります。サーバー構成パラメーターの場合は、START DATABASE MANAGER コマンドの実行時に有効になります。クライアント構成パラメーターの場合は、アプリケーションの再始動時に有効になります。

現行のデータベース・マネージャー構成パラメーターを表示したり印刷したりするには、 GET DATABASE MANAGER CONFIGURATION コマンドを使用してください。

関連概念:

- 「*管理ガイド: パフォーマンス*」の『ベンチマーク・テスト』

関連タスク:

- 168 ページの『複数の区画でのデータベース構成の変更』
- 「*管理ガイド: パフォーマンス*」の『構成パラメーターによる DB2 の構成』

関連資料:

- 「*コマンド・リファレンス*」の『GET DATABASE MANAGER CONFIGURATION コマンド』
- 「*コマンド・リファレンス*」の『UPDATE DATABASE MANAGER CONFIGURATION コマンド』

複数の区画でのデータベース構成の変更

手順:

複数の区画にわたってパーティション化されたデータベースを持っている場合、データベース構成ファイルは、すべてのデータベース・パーティションで同じものである必要があります。 SQL コンパイラーは、ノードの構成ファイルの情報に基づいて分散 SQL ステートメントをコンパイルし、 SQL ステートメントのニーズを満足させるためのアクセス・プランを作成するので、これらの構成ファイルには整合性が必要です。 データベース・パーティションごとに異なる構成ファイルを維持していると、どのデータベース・パーティションでステートメントが準備されたかによって、異なるアクセス・プランが作成される可能性があります。 **db2_all** を使用して、すべてのデータベース・パーティションで構成ファイルを保守してください。

関連概念:

- 379 ページの『パーティション・データベース環境でのコマンドの実行』

関連タスク:

- 166 ページの『ノードおよびデータベース構成ファイルの変更』

データベースの変更

データベースを変更するときのタスクは、データベースを作成するときのタスクと同じほど多くあります。これらのタスクは、以前に作成されたデータベースのある性質を更新したりドロップしたりします。タスクには以下のものがあります。

- 169 ページの『データベースのドロップ』
- 170 ページの『データベース・パーティション・グループの変更』

- 170 ページの『表スペースの変更』
- 180 ページの『スキーマのドロップ』
- 183 ページの『第 6 章 表および関連する表オブジェクトの変更』
- 205 ページの『ユーザー定義構造型タイプの変更』
- 206 ページの『型付き表の行の削除および更新』
- 206 ページの『既存の表または索引の名前変更』
- 209 ページの『表のドロップ』
- 210 ページの『ユーザー定義の一時表のドロップ』
- 211 ページの『トリガーのドロップ』
- 212 ページの『ユーザー定義関数 (UDF)、関数マッピング、またはメソッドのドロップ』
- 212 ページの『ユーザー定義タイプ (UDT) またはタイプ・マッピングのドロップ』
- 213 ページの『ビューの変更またはドロップ』
- 215 ページの『作動不能ビューの回復』
- 216 ページの『マテリアライズ照会またはステージング表のドロップ』
- 217 ページの『作動不能サマリー表の回復』
- 217 ページの『索引、索引の拡張、または索引の指定のドロップ』
- 219 ページの『オブジェクトを変更するときのステートメント従属関係』

データベースのドロップ

手順:

データベース内の一部のオブジェクトは変更可能ですが、データベースそのものを変更することはできません。変更するためには、データベースをドロップして再作成しなければなりません。データベースをドロップすることは、データベース内のすべてのオブジェクト、コンテナー、関連ファイルが削除されるため、広範囲に及ぶ影響があります。データベースをドロップすると、そのデータベースはデータベース・ディレクトリーからドロップ (アンカタログ) されます。

コントロール・センターを使用してデータベースをドロップするには、以下のようにします。

1. オブジェクト・ツリーを順に展開し、**「データベース (Databases)」**フォルダーを表示します。
2. ドロップしたいデータベースを右クリックして、ポップアップ・メニューから**「ドロップ (Drop)」**を選択します。
3. **「確認 (Confirmation)」**ボックスをクリックして、**「OK」**をクリックします。

コマンド行を使用してデータベースをドロップするには、以下のように入力します。

```
DROP DATABASE <name>
```

次のコマンドは、SAMPLE というデータベースを削除するものです。

```
DROP DATABASE SAMPLE
```

注: SAMPLE データベースについての実験を続けるつもりであるなら、このデータベースはドロップしないでください。 SAMPLE データベースをドロップした後で再び必要であることが分かった場合には、これを再作成できます。

関連資料:

- 「コマンド・リファレンス」の『GET SNAPSHOT コマンド』
- 「コマンド・リファレンス」の『DROP DATABASE コマンド』
- 「コマンド・リファレンス」の『LIST ACTIVE DATABASES コマンド』

データベース・パーティション・グループの変更

手順:

パーティションを追加またはドロップしたら、データベース・パーティション・グループ内の新しいパーティションのセットにわたって、現行データを再配分しなければなりません。これを行うためには、REDISTRIBUTE DATABASE PARTITION GROUP コマンドを使用します。

関連概念:

- 「*管理ガイド: パフォーマンス*」の『データ再配分』
- 「*管理ガイド: パフォーマンス*」の『データベース・サーバー容量の管理』

関連タスク:

- 「*管理ガイド: パフォーマンス*」の『パーティション間でのデータの再分散』

関連資料:

- 「コマンド・リファレンス」の『REDISTRIBUTE DATABASE PARTITION GROUP コマンド』

表スペースの変更

手順:

データベースを作成するときに、少なくとも 3 つの表スペースを作成します。それらの表スペースは、カタログ表スペース (SYSCATSPACE)、ユーザー表スペース (デフォルトの名前は USERSPACE1)、および SYSTEM TEMPORARY 表スペース (デフォルトの名前は TEMPSPACE1) です。これらの表スペースのそれぞれのうちの少なくとも 1 つは保持しなければなりません。必要に応じて、さらにユーザー表スペースと TEMPORARY 表スペースを余分に追加することができます。

注: カタログ表スペース SYSCATSPACE はドロップできませんし、もう 1 つ作成することもできません。また、常に少なくとも 4 KB のページ・サイズを持つ 1 つの SYSTEM TEMPORARY 表スペースが必要です。その他の SYSTEM TEMPORARY 表スペースは作成できます。表スペースのページ・サイズやエクステント・サイズを、作成後に変更することもできません。

関連タスク:

- 171 ページの『DMS 表スペースへのコンテナーの追加』
- 172 ページの『DMS 表スペース内のコンテナーの変更』

- 176 ページの『パーティション内の SMS 表スペースへのコンテナーの追加』
- 177 ページの『表スペースの名前変更』
- 178 ページの『ユーザー表スペースのドロップ』
- 179 ページの『SYSTEM TEMPORARY 表スペースのドロップ』
- 180 ページの『USER TEMPORARY 表スペースのドロップ』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『ALTER TABLESPACE ステートメント』

表スペースの変更に関する詳細

このセクションでは、表スペースの変更に関連したタスクを検討します。

DMS 表スペースへのコンテナーの追加

手順:

DMS 表スペース (つまり、MANAGED BY DATABASE 文節を使用して作成されたもの) のサイズは、 1 つ以上のコンテナーを表スペースに追加することによって増やすことができます。

表スペースに新しいコンテナーを追加したり、既存のコンテナーを拡張したりすると、表スペースの再平衡化という処理が発生する場合があります。バランスの再調整のプロセスには、表スペース・エクステントをある場所から別の場所に移動することが含まれます。このプロセスの最中、データは表スペース内にストライピングされた状態に保たれます。バランスの再調整は、必ずしもすべてのコンテナーにわたって発生するわけではなく、既存のコンテナー構成、新規コンテナーのサイズ、表スペースがどの程度満杯になっているかなど、多くの要素に依存しています。

コンテナーが既存の表スペースに追加される時には、コンテナーがストライプ 0 で開始されないように追加されます。マップ内のどこで開始するかはデータベース・マネージャーにより決定され、それは追加されるコンテナーのサイズに基づいてなされます。追加されるコンテナーが十分に大きくなければ、そのコンテナーがマップの最後のストライプで終わるように配置されます。それが十分に大きい場合には、ストライプ 0 で開始するように配置されます。

新規コンテナーを追加し、新規ストライプ・セットを作成する場合には、バランスの再調整は発生しません。 ALTER TABLESPACE ステートメントで BEGIN NEW STRIPE SET 文節を使用して、新規ストライプ・セットを作成します。さらに、ALTER TABLESPACE ステートメントで ADD TO STRIPE SET 文節を使用して、既存のストライプ・セットにコンテナーを追加することもできます。

バランスの再調整中も、表スペースへのアクセスは制限されません。複数のコンテナーを追加する必要がある場合は、それらを同時に追加しなければなりません。

コントロール・センターを使用して DMS 表スペースをコンテナーに追加するには、以下のようにします。

1. オブジェクト・ツリーを順に展開し、**「表スペース (Table Spaces)」**フォルダーを表示します。
2. コンテナーに追加したい表スペースを右クリックし、ポップアップ・メニューから**「更新 (Alter)」**を選択します。
3. **「追加 (Add)」**をクリックし、情報をすべて入力して、**「OK」**をクリックします。

コマンド行を使用して DMS 表スペースをコンテナーに追加するには、以下のようにします。

```
ALTER TABLESPACE <name>
   ADD (DEVICE '<path>' <size>, FILE '<filename>' <size>)
```

以下の例は、UNIX ベースのシステム上にある表スペースに対して、 2 つの新しいデバイス・コンテナー (それぞれが 10 000 ページのもの) を追加する方法を示したものです。

```
ALTER TABLESPACE RESOURCE
  ADD (DEVICE '/dev/rhd9'  10000,
       DEVICE  '/dev/rhd10' 10000)
```

ALTER TABLESPACE ステートメントを使用すると、パフォーマンスに影響を及ぼす可能性のある、表スペースの他の特性を変更することができます。

関連概念:

- 「*管理ガイド: パフォーマンス*」の『表スペースが照会の最適化に与える影響』
- 「*管理ガイド: プランニング*」の『DMS 表スペースでコンテナーを追加/拡張する方法』

関連タスク:

- 176 ページの『パーティション内の SMS 表スペースへのコンテナーの追加』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『ALTER TABLESPACE ステートメント』

DMS 表スペース内のコンテナーの変更

制約事項:

それぞれのロー・デバイスは、1 つのコンテナーとしてのみ使用できます。ロー・デバイスのサイズは、それが作成された後は固定されます。サイズ変更や拡張オプションを使用してロー・デバイス・コンテナーを増やすことを考慮している時には、まずロー・デバイスのサイズをチェックして、デバイス・コンテナーのサイズをロー・デバイスのサイズより大きく増やそうとすることが決してないようにしなければなりません。

手順:

DMS 表スペース (つまり、MANAGED BY DATABASE 文節を使用して作成されたもの) のコンテナーをサイズ変更することができます。

コントロール・センターを使用して DMS 表スペース中の 1 つ以上のコンテナーを増やすには、以下のようにします。

1. オブジェクト・ツリーを順に展開し、**「表スペース (Table Spaces)」**フォルダーを表示します。
2. コンテナーに追加したい表スペースを右クリックし、ポップアップ・メニューから**「更新 (Alter)」**を選択します。
3. **「サイズ変更 (Resize)」**をクリックし、情報をすべて入力して、**「OK」**をクリックします。

さらに、DMS 表スペースから既存のコンテナーをドロップしたり、 DMS 表スペースにある既存のコンテナーのサイズを縮小したり、さらにすべてのコンテナーにわたるデータのバランスの再調整を必要とせずに DMS 表スペースに新規コンテナーを追加したりもできます。

既存の表スペース・コンテナーのドロップや既存のコンテナーのサイズの縮小が可能なのは、ドロップされるエクステントまたはサイズが縮小されるエクステントの数が、表スペースの最高水準点を超えた位置にあるフリー・エクステントの数以下である場合に限られます。この場合の最高水準点とは、表スペース内で割り振られている最高ページのページ番号です。このマークは、表スペースの使用済みページの数と同じではありません。最高水準点よりも低い位置にあるエクステントの中には、再利用のために入手できるものもあるからです。

最高水準点まで (それを含む) のエクステントはすべて、表スペース内の同じ論理位置にとどまっている必要があるため、表スペースの最高水準点より上にあるフリー・エクステントの数は重要です。その操作によって生成される表スペースには、データをすべて収容するだけのスペースが必要になります。十分のフリー・スペースがない場合には、エラー・メッセージ (SQL20170N, SQLSTATE 57059) が出されます。

コンテナーをドロップするには、ALTER TABLESPACE ステートメントで DROP オプションを使用します。たとえば、以下のとおりです。

```
ALTER TABLESPACE TS1 DROP (FILE 'file1', DEVICE '/dev/rdisk1')
```

既存のコンテナーのサイズを縮小するには、RESIZE オプションまたは REDUCE オプションを使用します。 RESIZE オプションを使用する時、ステートメントの一部としてリストされているすべてのコンテナーは、サイズを増加させるか、サイズを縮小させるかの一方しか行えません。同じステートメント内で、幾つかのコンテナーを増やして、他のコンテナーを減らすことはできません。コンテナーのサイズの新しい下限がわかっている場合には、サイズ変更方式を考慮する必要があります。コンテナーの現在のサイズがわからない場合 (および検討していない場合) には、縮小方式を考慮する必要があります。

コマンド行を使用して DMS 表スペース中の 1 つ以上のコンテナーのサイズを減らすには、以下のようにします。

```
ALTER TABLESPACE <name>
   REDUCE (FILE '<filename>' <size>)
```

以下の例は、Windows ベースのシステム上の表スペース中にある、 FILE コンテナー (すでに 1 000 ページあるもの) のサイズを縮小する方法を示したものです。

```
ALTER TABLESPACE PAYROLL
  REDUCE (FILE 'd:¥hldr¥finance'  200)
```

このアクションを実行すると、ファイルは 1 000 ページから 800 ページにサイズが減少します。

コマンド行を使用して DMS 表スペース中の 1 つ以上のコンテナーのサイズを増やすには、以下のようにします。

```
ALTER TABLESPACE <name>
   RESIZE (DEVICE '<path>' <size>)
```

以下の例は、UNIX ベースのシステム上の表スペース中にある、 2 つのデバイス・コンテナー (それぞれが 1 000 ページのもの) のサイズを増やす方法を示したものです。

```
ALTER TABLESPACE HISTORY
  RESIZE (DEVICE '/dev/rhd7'  2000,
          DEVICE  '/dev/rhd8' 2000)
```

このアクションを実行すると、 2 つのデバイスは 1 000 ページから 2 000 ページにサイズが増えます。すべてのコンテナーにわたって表スペースのコンテナーのバランスが再調整されます。バランスの再調整中も、表スペースへのアクセスは制限されません。

コマンド行を使用して DMS 表スペース中の 1 つ以上のコンテナーを拡張するには、以下のようにします。

```
ALTER TABLESPACE <name>
   EXTEND (FILE '<filename>' <size>)
```

以下の例は、Windows ベースのシステム上の表スペース中にある、ファイル・コンテナー (それぞれが 1 000 ページのもの) のサイズを増やす方法を示したものです。

```
ALTER TABLESPACE PERSNEL
  EXTEND (FILE 'e:¥wrkhist1'  200
          FILE  'f:¥wrkhist2'  200)
```

このアクションを実行すると、 2 つのファイルは 1 000 ページから 1 200 ページにサイズが増えます。すべてのコンテナーにわたって表スペースのコンテナーのバランスが再調整されます。バランスの再調整中も、表スペースへのアクセスは制限されません。

表スペースの作成中または作成後に追加される DMS コンテナー (ファイルとロー・デバイス・コンテナーの両方)、あるいは表スペースの作成中または作成後に拡張される DMS コンテナーは、プリフェッチャーを使用して並列で実行されます。このような作成またはサイズ変更コンテナー操作の並列処理を高めるために、システムで実行されるプリフェッチャーの数を増やすことができます。並列で行われないプロセスは、これらのアクションのロギングと、コンテナーの作成の場合は、コンテナーのタグ付けだけです。

注: CREATE TABLESPACE または ALTER TABLESPACE ステートメントの並列処理 (新しいコンテナーの既存の表スペースへの追加に関して) を最大限にするには、プリフェッチャーの数が、追加されるコンテナーの数と等しいかそれ以上であることを確かめてください。プリフェッチャーの数は、*num_ioservers* データベース構成パラメーターにより制御されます。データベースは、新規パラ

メーター値を有効にするためには、停止しなければなりません。つまり、すべてのアプリケーションおよびユーザーは、変更を有効にするためにデータベースから切断する必要があります。

ALTER TABLESPACE ステートメントを使用すると、パフォーマンスに影響を及ぼす可能性のある、表スペースの他の特性を変更することができます。

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『ALTER TABLESPACE ステートメント』

コンテナーを追加またはドロップした後の自動プリフェッチ・サイズ調整

コンテナーを追加またはドロップした後に表スペースのプリフェッチ・サイズの更新を忘れる可能性がある場合、データベース・マネージャーがプリフェッチ・サイズを自動的に決めるようにすることを考慮できます。プリフェッチ・サイズの更新を忘れると、データベースのパフォーマンスが大幅に低下することがあります。

DB2® Universal Database (DB2 UDB) は、バージョン 8.2 (またはそれ以降) で作成した表スペースのデフォルトが自動プリフェッチ・サイズになるようにセットアップされています。データベース・マネージャーは以下の公式を使用して、表スペースのプリフェッチ・サイズを計算します。

```
プリフェッチ・サイズ = (コンテナーの数) X (コンテナーごとの物理スピンドル
                       の数) X エクステント・サイズ
```

表スペースのプリフェッチ・サイズを AUTOMATIC に設定しない方法には、以下の 3 つがあります。

- 特定のプリフェッチ・サイズで表スペースを作成する。プリフェッチ・サイズの値を手動で選択するということは、表スペースに関連したコンテナーの数が調整されたときにはいつでも、自分で忘れずにプリフェッチ・サイズを調整するということです (調整が必要な場合)。
- 表スペースを作成する際にはプリフェッチ・サイズを使用せず、 *dft_prefetch_sz* データベース構成パラメーターを AUTOMATIC ではない値に設定する。表スペースを作成するときにプリフェッチ・サイズを明示的に指定しない場合、DB2 UDB はこのパラメーターを調べます。 AUTOMATIC 以外の値が検出されると、その値がデフォルトのプリフェッチ・サイズとして使用されます。表スペースに関連したコンテナーの数が調整されたときにはいつでも、自分で忘れずにプリフェッチ・サイズを調整しなければなりません (調整が必要な場合)。
- ALTER TABLESPACE ステートメントを使用して、プリフェッチ・サイズを手動で変更する。

DB2_PARALLEL_IO の使用

プリフェッチ要求は、表スペースの並列処理に基づいて、要求がプリフェッチ・キューにサブミットされる前に、複数の小さなプリフェッチ要求に分割されます。コンテナーごとの物理スピンドル数を定義し、表スペース上の並列入出力に影響を与えるために、DB2_PARALLEL_IO レジストリー変数が使用されます。並列入出力がオフであると、表スペースの並列処理はコンテナーの数と等しくなります。並列入出力がオンであると、表スペースの並列処理は、コンテナーの数に DB2_PARALLEL_IO レジストリー変数で指定した値を乗算した結果と等しくなりま

す。 (つまり、表スペースの並列処理は、プリフェッチ・サイズを表スペースのエクステント・サイズで割った値と等しくなります。)

以下の例は、 DB2_PARALLEL_IO レジストリー変数がプリフェッチ・サイズにどのように影響を与えるかを示しています。 (以下の表スペースはすべて、AUTOMATIC プリフェッチ・サイズが指定されているものと想定してください。)

- DB2_PARALLEL_IO=*
  - すべての表スペースはデフォルト (各コンテナーのスピンドル数が 6) を使用します。並列入出力がオンであると、プリフェッチ・サイズは 6 倍大きくなります。
  - すべての表スペースでは、並列入出力がオンになります。プリフェッチ要求は複数の小さな要求に分割されます。それぞれの要求は、プリフェッチ・サイズをエクステント・サイズで割った値と等しくなります (または、コンテナー数にスピンドル数を乗じた値に等しくなります)。
- DB2_PARALLEL_IO=*:3
  - すべての表スペースは、コンテナーごとのスピンドル数として 3 を使用します。
  - すべての表スペースでは、並列入出力がオンになります。
- DB2_PARALLEL_IO=*:3,1:1
  - すべての表スペースはコンテナーごとのスピンドル数として 3 を使用しますが、表スペース 1 は例外で 1 を使用します。
  - すべての表スペースでは、並列入出力がオンになります。

関連タスク:

- 170 ページの『表スペースの変更』
- 171 ページの『DMS 表スペースへのコンテナーの追加』
- 172 ページの『DMS 表スペース内のコンテナーの変更』

パーティション内の SMS 表スペースへのコンテナーの追加

制約事項:

コンテナーを追加できるのは、現在コンテナーがないパーティション上の SMS 表スペースに対してのみです。

手順:

コマンド行を使用して、SMS 表スペースにコンテナーを追加するには、以下のように入力します。

```
ALTER TABLESPACE <name>
   ADD ('<path>')
   ON DBPARTITIONNUM (<partition_number>)
```

番号で指定されたパーティション、およびパーティションの範囲内のすべてのパーティション (もしくはノード) は、表スペースが定義されているデータベース・パーティション・グループに存在していなくてはなりません。 partition_number は、ステートメントのただ 1 つの *db-partitions-clause* で、明示的にのみ、または範囲で表される場合もあります。

以下の例では、UNIX ベースのオペレーティング・システム上で、表スペース「plans」が使用しているデータベース・パーティション・グループの 3 番パーティションにどのように新規のコンテナーを追加するかを示しています。

```
ALTER TABLESPACE plans
   ADD ('/dev/rhdisk0')
   ON DBPARTITIONNUM (3)
```

関連タスク:

- 171 ページの『DMS 表スペースへのコンテナーの追加』
- 172 ページの『DMS 表スペース内のコンテナーの変更』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『ALTER TABLESPACE ステートメント』

表スペースの名前変更

制約事項:

SYSCATSPACE 表スペースの名前を変更することはできません。

「ロールフォワード・ペンディング」または「ロールフォワード進行中」状態の表スペースの名前を変更することはできません。

バックアップを取った後に名前変更した表スペースをリストアする際には、RESTORE DATABASE コマンド中で新しい表スペースの名前を使用しなければなりません。変更前の表スペース名を使用しても、検出されません。同様に、ROLLFORWARD DATABASE コマンドを使用して表スペースをロールフォワードする場合も、必ず新しい名前を使用するようにしてください。変更前の表スペース名を使用しても、検出されません。

手順:

既存の表スペースに、その中の個々のオブジェクトとは関係のない名前を新たに付けることができます。表スペースの名前を変更すると、その表スペースを参照するカタログ・レコードもすべて変更されます。

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『RENAME TABLESPACE ステートメント』

表スペース状態の切り替え

手順:

表スペースに関連付けられたコンテナーがすでにアクセス可能になっている場合、ALTER TABLESPACE ステートメントの SWITCH ONLINE 文節を使用して、表スペースから OFFLINE 状態を除去できます。表スペースは、データベースがまだ起動し作動している間に、OFFLINE 状態を除去します。

この文節を使用する代わりに、データベースからすべてのアプリケーションを切断し、再びアプリケーションをデータベースに接続しなおすこともできます。これで、表スペースから OFFLINE 状態が除去されます。

コマンド行を使用して表スペースから OFFLINE 状態を除去するには、以下のように入力します。

```
db2 ALTER TABLESPACE <name>
   SWITCH ONLINE
```

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『ALTER TABLESPACE ステートメント』

ユーザー表スペースのドロップ

手順:

ユーザー表スペースをドロップするときには、その表スペース内のすべてのデータを削除し、コンテナーを解放し、カタログ項目をドロップし、そして、その表スペースの中に定義されたすべてのオブジェクトをドロップするか無効のマークを付けます。

表スペースをドロップすることによって、空の表スペース内のコンテナーを再使用することができますが、コンテナーを再使用する前に、DROP TABLESPACE コマンドを COMMIT しなければなりません。

該当するシングル・ユーザー表スペース内の索引および LOB データを含むすべての表データが入ったユーザー表スペースをドロップできます。また、いくつかの表スペースにわたる表を持つようなユーザー表スペースもドロップできます。つまり、1 つの表スペースに表データ、別の表スペースに索引、 3 番目の表スペースに LOB を持つ場合があります。あるいは、単一のステートメントで 3 つの表スペースすべてを同時にドロップする必要があります。 スパンされる表が入った表スペースはすべて、この単一ステートメントの一部にする必要があります。さもなければ、ドロップ要求は失敗します。

コントロール・センターを使用してユーザー表スペースをドロップするには、以下のようにします。

1. オブジェクト・ツリーを順に展開し、**「表スペース (Table Spaces)」**フォルダーを表示します。
2. ドロップしたい表スペースを右クリックして、ポップアップ・メニューから**「ドロップ (Drop)」**を選択します。
3. **「確認 (Confirmation)」**ボックスにチェックを付け、**「OK」**をクリックします。

コマンド行を使用してユーザー表スペースをドロップするには、以下のように入力します。

```
DROP TABLESPACE <name>
```

以下の SQL ステートメントは、表スペース ACCOUNTING をドロップします。

```
DROP TABLESPACE ACCOUNTING
```

関連タスク:

- 179 ページの『SYSTEM TEMPORARY 表スペースのドロップ』
- 180 ページの『USER TEMPORARY 表スペースのドロップ』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『COMMIT ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『DROP ステートメント』

SYSTEM TEMPORARY 表スペースのドロップ

制約事項:

まず SYSTEM TEMPORARY 表スペースをもう 1 つ作成してからでなければ、 4 KB のページ・サイズを持つ SYSTEM TEMPORARY 表スペースをドロップすることはできません。データベースは常に少なくとも 1 つの 4 KB のページ・サイズを持つ SYSTEM TEMPORARY 表スペースを持っていなければならないため、新規 SYSTEM TEMPORARY 表スペースは、4 KB のページ・サイズを持っている必要があります。たとえば、4 KB のページ・サイズを持つ単一 SYSTEM TEMPORARY 表スペースを持っていて、そこにコンテナーを追加したいと思っており、その表スペースが SMS 表スペースであるならば、まず適切なコンテナー数を持つ新しい 4 KB のページ・サイズの SYSTEM TEMPORARY 表スペースを追加してから、古い SYSTEM TEMPORARY 表スペースをドロップしなければなりません。 (DMS を使用している場合には、表スペースをドロップして再作成することなく、コンテナーを追加することができます。)

手順:

デフォルトの表スペース・ページ・サイズは 4 KB です。

コントロール・センターを使用してシステム表スペースをドロップするには、以下のようにします。

1. オブジェクト・ツリーを順に展開し、**「表スペース (Table Spaces)」**フォルダーを表示します。
2. 他に SYSTEM TEMPORARY 表スペースが 1 つしかない場合は、**「表スペース (Table Spaces)」**フォルダーを右クリックして、ポップアップ・メニューから**「作成 (Create)」—>「ウィザードを使用する表スペース (Table Spaces Using Wizard)」**を選択します。それ以外の場合は、スキップ 4 にスキップします。
3. このウィザードのステップに従って、必要に応じて新しい SYSTEM TEMPORARY 表スペースを作成します。
4. **「表スペース (Table Spaces)」**フォルダーを再度クリックし、ウィンドウ (「内容 (Contents)」ペイン) の右側に表スペースのリストを表示します。
5. ドロップしたい SYSTEM TEMPORARY 表スペースを右クリックして、ポップアップ・メニューから**「ドロップ (Drop)」**をクリックします。
6. **「確認 (Confirmation)」**ボックスにチェックを付け、**「OK」**をクリックします。

次は、SYSTEM TEMPORARY 表スペースを作成するためのステートメントです。

```
CREATE SYSTEM TEMPORARY TABLESPACE <name>
   MANAGED BY SYSTEM USING ('<directories>')
```

続いて、コマンド行を使用してシステム表スペースをドロップするには、以下のように入力します。

```
DROP TABLESPACE <name>
```

以下の SQL は、TEMPSPACE2 と呼ばれる新しい SYSTEM TEMPORARY 表スペースを作成します。

```
CREATE SYSTEM TEMPORARY TABLESPACE TEMPSPACE2
   MANAGED BY SYSTEM USING ('d:¥systemp2')
```

TEMPSPACE2 が作成されれば、以下のコマンドを使用して、元の SYSTEM TEMPORARY 表スペース TEMPSPACE1 をドロップすることができます。

```
DROP TABLESPACE TEMPSPACE1
```

表スペースをドロップすることによって、空の表スペース内のコンテナーを再使用することができますが、コンテナーを再使用する前に、 DROP TABLESPACE コマンドを COMMIT しなければなりません。

関連タスク:

- 178 ページの『ユーザー表スペースのドロップ』
- 180 ページの『USER TEMPORARY 表スペースのドロップ』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『CREATE TABLESPACE ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『DROP ステートメント』

USER TEMPORARY 表スペースのドロップ

手順:

USER TEMPORARY 表スペースをドロップできるのは、その表スペースに現行で定義されている、宣言済み一時表がない場合に限ります。表スペースをドロップするときに、その表スペース内の宣言済み一時表のドロップが試みられることはまったくありません。

注: 宣言済み一時表を宣言しているアプリケーションがデータベースから切断されると、宣言済み一時表は暗黙的にドロップされます。

関連タスク:

- 178 ページの『ユーザー表スペースのドロップ』
- 179 ページの『SYSTEM TEMPORARY 表スペースのドロップ』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『DROP ステートメント』

スキーマのドロップ

手順:

スキーマをドロップする前に、そのスキーマの中にあったオブジェクト自体をすべてドロップするか、別のスキーマに移動しなければなりません。 DROP ステートメントを試みているときには、スキーマ名がカタログの中になければなりません。そうでない場合、エラーが戻されます。

コントロール・センターを使用してスキーマをドロップするには、以下のようにします。

1. オブジェクト・ツリーを順に展開し、**「スキーマ (Schemas)」**フォルダーを表示します。
2. ドロップしたいスキーマを右クリックして、ポップアップ・メニューから**「ドロップ (Drop)」**を選択します。
3. **「確認 (Confirmation)」**ボックスにチェックを付け、**「OK」**をクリックします。

コマンド行を使用してスキーマをドロップするには、以下のように入力します。

```
DROP SCHEMA <name>
```

以下の例では、″joeschma″ というスキーマがドロップされます。

```
DROP SCHEMA joeschma RESTRICT
```

RESTRICT キーワードは、データベースから削除するよう指定されたスキーマにはオブジェクトを定義できないという規則を、強制的に実行します。

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『DROP ステートメント』

バッファー・プールの変更

既存のバッファー・プールを使用する際には、以下のいずれかのタスクの実行が必要になる場合があります。

- すべてのパーティション、または単一のパーティションにおける、バッファー・プールのサイズの変更。
- 拡張ストレージの使用可能化または使用不能化。
- 新規データベース・パーティション・グループへのこのバッファー・プール定義の追加。
- ブロック・ベース入出力のための、バッファー・プールのブロック領域の変更。

前提条件:

このステートメントの許可 ID には、SYSCTRL 権限または SYSADM 権限がなければなりません。

手順:

1. SELECT BPNAME FROM SYSCAT.BUFFERPOOLS を実行し、データベース内に既に存在するバッファー・プール名をリストします。
2. 結果として表示されるリストからバッファー・プール名を選択します。
3. どのような変更を行う必要があるかを判断します。
4. ALTER BUFFERPOOL ステートメントを実行する適正な許可 ID があることを確認します。

   注: 2 つのキー・パラメーターは IMMEDIATE と DEFERRED です。 IMMEDIATE では、バッファー・プール・サイズが即時に変更されます。データベース共用メモリーに、新しいスペースを割り振るための十分な予約スペースがない場合、ステートメントの実行は保留にされます。

DEFERRED の場合、バッファー・プールは、アプリケーションの接続が切断され、データベースが再活動化されるときにキャッシュに入れられます。予約済みのメモリー・スペースは必要ありません。活動化の際に、DB2 UDB が、必要なメモリーをシステムから割り振ります。

5. ALTER BUFFERPOOL ステートメントを使用して、バッファー・プール・オブジェクトの単一品質を変更します。

関連タスク:

• 74 ページの『バッファー・プールの作成』

関連資料:

• 「*SQL リファレンス 第 2 巻*」の『ALTER BUFFERPOOL ステートメント』

第 6 章 表および関連する表オブジェクトの変更

表および関連する表オブジェクトの構造と内容を修正するために必要なタスクには、次の事柄が含まれます。

- 『既存の表のスペース圧縮』
- 187 ページの『既存の表への列の追加』
- 187 ページの『列定義の修正』
- 189 ページの『表またはビューからの行の除去』
- 208 ページの『MERGE ステートメントを使用して、表およびビューの内容を更新する』
- 190 ページの『ID 列定義の変更』
- 191 ページの『制約の変更』
- 197 ページの『既存の表での生成列の定義』
- 200 ページの『表の揮発性を宣言する』
- 201 ページの『パーティション・キーの変更』
- 202 ページの『表属性の変更』
- 204 ページの『マテリアライズ照会表のプロパティーの変更』
- 205 ページの『マテリアライズ照会表のデータのリフレッシュ』

表に対するトリガーは変更できないことに注意してください。適切でなくなったトリガーをドロップして (211 ページの『トリガーのドロップ』を参照)、その代わりのものを追加 (123 ページの『トリガーの作成』を参照) しなければなりません。

既存の表および関連する表オブジェクトの変更

既存の表のスペース圧縮

既存の表は、スペース圧縮を許可するレコード・フォーマットに変更できます。スペース圧縮を許可するレコード・フォーマットにある列のバイト・カウントの合計は、表スペース内の表で許容される行の長さを超えないかぎり、元の (スペース圧縮を許可しない) レコード・フォーマットにある列のバイト・カウントの合計を超えるかもしれません。たとえば、許容される行の長さは、4 KB のページ・サイズを持つ表スペースでは 4005 バイトです。許容される行の長さを超える場合には、エラー・メッセージ SQL0670N が戻ります。バイト・カウントの公式は、CREATE TABLE ステートメントの一部として文書化されています。

同様に、既存の表は、スペース圧縮を許可するレコード・フォーマットから、許可しないレコード・フォーマットに変更できます。列のバイト・カウントの合計に関して同じ条件が適用され、エラー・メッセージ SQL0670N が必要に応じて戻されます。

表のスペース圧縮を考慮すべきかどうかを判断するために、ほとんどがシステム・デフォルト値に等しい値、または NULL になっている表は、新しく行フォーマット

を行う利点があるということを知っておくとよいでしょう。たとえば、 INTEGER 列と 90% の列が 0 の値 (データ・タイプ INTEGER のデフォルト値) または NULL になっているところでは、この表に加えてこの列を圧縮すると、新しく行フォーマットを行う利点が得られ、多くのディスク・スペースを節約します。

表を変更する時には、VALUE COMPRESSION 文節を使用して、表が表レベルで (おそらく列レベルでも) スペース行フォーマットを使用しているということを指定することができます。 ACTIVATE VALUE COMPRESSION を使用して、表がスペース節約手法を使用することを指定するか、または DEACTIVATE VALUE COMPRESSION を使用して、表がその表のデータではもはやスペース節約手法を使用しないことを指定します。

DEACTIVATE VALUE COMPRESSION を使用する場合は、そのことにより、その表の列に関連したすべての COMPRESS SYSTEM DEFAULT オプションを暗黙的に使用不可にします。

表を新しい行フォーマットに変更した後は、それ以降に挿入、ロード、または更新されるすべての行は、新しい行フォーマットを持ちます。すべての行を新しい行フォーマットに変更するために、行フォーマットを変更する前に、表の再編成を実行するか、既存の行に更新操作を行う必要があります。

関連概念:

- 100 ページの『新しい表のスペース圧縮』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『ALTER TABLE ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『CREATE TABLE ステートメント』

ストアード・プロシージャーを使った表の変更

表は、業務のデータすべてを保管する場所です。データベースを作成する前に、データベースに保管したいデータのタイプおよび編成を検討する必要があります。自分と自分の業務に必要となる可能性のある関連データすべてを使用して取り扱うよう、確実に考えておくには、多くの計画が必要です。しかし、物事は変化します。良い計画を立てても、新規の要件またはビジネス上の変更が生じて、データベース内の表を変更しなければならなくなることがあります。

表の中で、以下の 1 つ以上の方法で変更を加える必要があることに気付くことがあります。

- 列の名前変更
- 列の除去
- SQL スカラー関数を使用した、列タイプの変更および既存データのトランスフォーム
- 列サイズの増加または減少
- 列のデフォルト値の変更
- 列を NOT NULL から NULLABLE に変更
- 精度および小数部の位取りの変更

これらのタイプの変更を行うとき、元の表データを消失するリスクを最小にする必要があります。 DB2® Universal Database (DB2 UDB) は、表の変更を可能にする、ユーザー・インターフェースおよびストアード・プロシージャーを提供します。表を変更する作業のすべてが完了したことを明示的に示すまで、元の表および関連データはドロップされません。

ユーザー・インターフェースから呼び出される各ストアード・プロシージャー呼び出しは、上にリストしたアクションを達成するために、ドロップ、再作成、データのロードなどの一連のアクションを実行します。

表の中で何を変更できるかに関しては、制限があります。これらの制限には、以下のものがあります。

- マテリアライズ照会表 (MQT) の変更はサポートされていない。

  ただし、MQT を持つ表の変更はサポートされています。さらに、変更された基本表に定義されている MQT は、 ALTER TABLE プロセスの際に更新 (データを追加) されません。 MQT の基本表が ALTOBJ() ストアード・プロシージャーによって変更されている間、基本表から選択された結果の一部ではない列はすべて消失します。というのは、MQT の内容は、新しい基本表から完全に再作成されるからです。
- 型付き表、つまり既存の参照列タイプの表の有効範囲となる表の変更はサポートされていない。
- ニックネームを使用したリモート表の変更はサポートされていない。
- 表内の列の順序は再配列できない。
- 追加および名前変更は、列のドロップのアクション専用。

  つまり、これらの列アクションは、単一の表変更の呼び出しの中に共存できません。
- DATALINK データ・タイプはサポートされていない。
- 持続するオブジェクト・ロックはないので、オブジェクトの定義は複数の ALTOBJ() 呼び出しの間に変化することがある。
- テーブル・パック記述子に関連した runstats プロファイルなどの表プロファイルは、 ALTER TABLE プロセスが実施された後に消失する。
- 一度に 1 つの表に対してサポートされる ALTER TABLE ストアード・プロシージャー呼び出しのシーケンスは 1 つだけ。つまり、ALTOBJ() ストアード・プロシージャーが呼び出された後は、それが終了するかロールバックされてからでなければ、同じ表で他の ALTER TABLE を開始できません。 ALTOBJ() ストアード・プロシージャーを使用して複数の表を同時に変更することは、表の従属関係が競合しない限りサポートされています。

ALTER TABLE アクションを実行するストアード・プロシージャーを使用するときに、使用可能なオプションを構成する複数のコンポーネント部分があります。それらの部分は、以下のとおりです。

- ALTER_OBJ('GENERATE','<sql statement>', 0, ?)

  このプロシージャーはすべての SQL ステートメントを生成して、それらをメタデータ表に入れます。

注: 生成モードでは、SQL ステートメント・パラメーターは NULL であってはなりません。変更 ID が指定された場合、それは無視されます。

- ALTER_OBJ('VALIDATE',NULL,123,?)

  このプロシージャーは生成された SQL を検査しますが、データの移動は行われません。妥当性を検査するためのスクリプトの実行は、指定したユーザー ID「123」の下で行われます。検査の結果は、メタ表に入れられます (そこには変更される表からの他の情報も保管されています)。

- ALTER_OBJ('APPLY_CONTINUE_ON_ERROR',NULL,123,?)

  このプロシージャーは、指定の ID の下にあるすべての SQL ステートメントを実行して、その結果をメタ表に書き込みます。 SQL ステートメントには、新規の表を作成する方法、従属オブジェクトを作成すること、および新規の表にデータを追加することが含まれます。

  UNDO モードを使用して、元の定義に戻すことができます (下記を参照)。

  警告 SQLCODE が SQLCA 内のストアード・プロシージャーに対して設定されて、ストアード・プロシージャー内のトランザクションが終了します。

- ALTER_OBJ('APPLY_STOP_ON_ERROR',NULL,123,?)

  このプロシージャーは、指定の ID の下にある個々の SQL ステートメントを 1 つずつ実行して、エラーを検出したときには停止します。

  エラー SQLCODE が SQLCA 内のストアード・プロシージャーに対して設定されて、ストアード・プロシージャー内のトランザクションが自動的にロールバックします。

- ALTER_OBJ('UNDO',NULL,123,?)

  指定のユーザー ID による表変更アクションによって行われた、すべての変更を含むスクリプトを実行します。それらの変更は、すべて取り消されます。

  注: ALTOBJ_UNDO に対して作業するときは、ID パラメーターを NULL にすることはできません。

- ALTER_OBJ('FINISH',NULL,123,?)

  このプロシージャーは、元の表を削除して、指定のユーザー ID の下にあるメタ表のすべての項目をクリーンアップします。

  注: このモードは、他のすべてのモードとは別に呼び出す必要があります。

関連資料:

- 「*SQL リファレンス 第 1 巻*」の『サポートされている関数と SQL 管理ルーチン』
- 「*SQL Administrative Routines*」の『ALTOBJ プロシージャー』

既存の表への列の追加

手順:

列定義には、列名、データ・タイプ、および必要な制約が含まれます。

表に列が追加されると、列は右端の既存の列定義の右側に論理的に配置されます。既存の表に新しい列を追加する場合、システム・カタログの表記述を修正するだけなので、表へのアクセス時間はすぐには影響を受けません。 UPDATE ステートメントによって修正される時点まで、既存のレコードが物理的に変更されることはありません。表から既存の行を取り出すと、新しい列には、列の定義にしたがって NULL 値かデフォルト値かが入れられます。表が作成された後に追加された列は、NOT NULL として定義することはできません。 NOT NULL WITH DEFAULT として、または NULL 値可能としてのいずれかで定義しなければなりません。

コントロール・センターを使用して既存の表に列を追加するには、以下のようにします。

1. オブジェクト・ツリーを順に展開し、**「表 (Tables)」**フォルダーを表示します。
2. 列を追加したい表を右クリックし、ポップアップ・メニューから**「更新 (Alter)」**を選択します。
3. **「列 (Columns)」**ページをチェックして、列に関する情報をすべて入力し、**「OK」**をクリックします。

コマンド行を使用して既存の表に列を追加するには、以下のようにします。

```
ALTER TABLE <table_name>
   ADD <column_name> <data_type> <null_attribute>
```

列の追加は、SQL ステートメントを使ってもできます。次のステートメントは、ALTER TABLE ステートメントを使って、 EMPLOYEE 表に 3 つの列を追加するものです。

```
ALTER TABLE EMPLOYEE
   ADD MIDINIT CHAR(1)   NOT NULL WITH DEFAULT
   ADD HIREDATE DATE
   ADD WORKDEPT CHAR(3)
```

関連タスク:

- 187 ページの『列定義の修正』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『ALTER TABLE ステートメント』

列定義の修正

手順:

既存の VARCHAR または VARGRAPHIC 列を長くして、列の特性を変更することができます。文字数は、使用されるページ・サイズに従属する値まで増やすことができます。

列に関連したデフォルト値は、変更することができます。いったん新しいデフォルト値を定義すると、デフォルトを使用することが指示されている後続のあらゆる SQL 操作で、その列の新しい値が使用されます。その新しい値は、割り当て規則に準拠している必要があり、 CREATE TABLE ステートメントに関して説明したものと同じ制約が課せられます。

注: 生成列のデフォルト値をこのステートメントで変更することはできません。

コントロール・センターを使用して既存の表の列の長さを修正するには、以下のようにします。

1. オブジェクト・ツリーを順に展開し、**「表 (Tables)」**フォルダーを表示します。
2. 右側のペイン内にある表のリストから、修正したい列を含む表を右クリックし、ポップアップ・メニューから**「更新 (Alter)」**を選択します。
3. **「列 (Columns)」**ページをチェックして、列を選択し、**「変更 (Change)」**をクリックします。
4. 列の新しいバイト・カウントを**「長さ (Length)」**に入力し、**「OK」**をクリックします。

コマンド行を使用して既存の表の列の長さおよびタイプを修正するには、以下のように入力します。

```
ALTER TABLE <table_name>
   ALTER COLUMN <column_name>
   <modification_type>
```

たとえば、列を 4000 文字まで増やすには、次のような形式で指定します。

```
ALTER TABLE t1
   ALTER COLUMN colnam1
   SET DATA TYPE VARCHAR(4000)
```

別の例として、列が新規の VARGRAPHIC 値を持つようにするには、次のような SQL ステートメントで指定します。

```
ALTER TABLE t1
   ALTER COLUMN colnam2
   SET DATA TYPE VARGRAPHIC(2000)
```

型付き表の列を変更することはできません。しかし、有効範囲がまだ定義されていない既存の参照タイプ列に、有効範囲を追加することは可能です。たとえば、以下のとおりです。

```
ALTER TABLE t1
   ALTER COLUMN colnamt1
   ADD SCOPE typtab1
```

コマンド行を使用して既存の表の列のデフォルト値を修正するには、以下のように入力します。

```
ALTER TABLE <table_name>
   ALTER COLUMN <column_name>
   SET DEFAULT 'new_default_value'
```

たとえば、列のデフォルト値を変更するには、次のような形式で指定します。

```
ALTER TABLE t1
   ALTER COLUMN colnam1
   SET DEFAULT '123'
```

関連タスク:

- 190 ページの『ID 列定義の変更』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『ALTER TABLE ステートメント』

表またはビューからの行の除去

手順:

行を削除することにより、表またはビューの内容を変更できます。ビューから行を削除すると、ビューの基本となっている表から行が削除されます。 DELETE ステートメントを使用して、次のことを行います。

- 検索条件によりオプションで判別されている 1 つ以上の行を削除する。これを*検索済み DELETE* と言います。
- カーソルの現在位置により判別されている 1 行だけを削除する。これを*位置決め DELETE* と言います。

DELETE ステートメントはアプリケーション・プログラムに組み込むことができ、また動的 SQL ステートメントとして出すことができます。

変更される表が参照制約によって他の表に関連している場合、行の削除を実行する際の考慮事項があります。識別された表、または識別されたビューの基本表が親である場合、削除するように選択した行は RESTRICT 削除規則に関連した従属関係を持っていてはなりません。さらに、DELETE は、RESTRICT 削除規則に関連した従属関係を持つ下層行にカスケードしてはなりません。

削除操作が RESTRICT 削除規則によって妨げられない場合、選択した行が削除されます。

たとえば、表 (DEPARTMENT) から部門 (DEPTNO) ″D11″ を削除するには、以下を使用します。

```
DELETE FROM department WHERE deptno='D11'
```

複数行の DELETE の実行中にエラーが発生する場合、表は変更されません。エラーの発生により、検索条件に一致するすべての行を削除できず、既存の参照制約で必要とされるすべての操作が妨げられる場合、表は変更されません。

適切なロックが存在していない場合、正常な DELETE ステートメントの実行時に、1 つ以上の排他ロックが獲得されます。 COMMIT または ROLLBACK ステートメントの後に、ロックが解放されます。ロックにより、他のアプリケーションは表に対して操作を実行できなくなります。

関連概念:

- 「*管理ガイド: パフォーマンス*」の『ロックおよび並行性の制御』
- 「*管理ガイド: パフォーマンス*」の『ロックとパフォーマンス』

- 「*管理ガイド: パフォーマンス*」の『ロッキングに影響を与える要因』
- 「*管理ガイド: パフォーマンス*」の『ロッキングのガイドライン』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『DELETE ステートメント』

列の生成または ID プロパティーの変更

ALTER TABLE ステートメントの ALTER COLUMN 文節を使用して、表の列の生成または ID プロパティーを追加およびドロップできます。

次のいずれかを行うことができます。

- 既存の非生成列に対して作業を行うとき、生成式属性を追加できます。その結果、変更された列は生成列となります。
- 既存の生成列に対して作業を行うとき、生成式属性をドロップできます。その結果、変更された列は通常の、非生成列となります。
- 既存の非 ID 列に対して作業を行うとき、ID 属性を追加できます。その結果、変更された列は ID 列となります。
- 既存の ID 列に対して作業を行うとき、ID 属性をドロップできます。その結果、変更された列は通常の、非生成、非 ID 列となります。
- 既存の生成列に対して作業を行うとき、生成列を GENERATED ALWAYS から GENERATED BY DEFAULT の状態に変更できます。その逆も可能です。つまり、生成列を GENERATED BY DEFAULT から GENERATED ALWAYS の状態に変更できます。これが可能なのは、生成列に対して作業を行うときだけです。
- ユーザー定義のデフォルト列からデフォルト属性をドロップできます。これを行うと、新規のデフォルト値は NULL となります。
- 同じ ALTER COLUMN ステートメントで、デフォルト、ID、または生成属性をドロップしてから、新規のデフォルト、ID、または生成属性を設定できます。
- CREATE TABLE および ALTER TABLE ステートメントの両方で、『ALWAYS』 は GENERATED 文節の中のオプションのワードです。つまり、ALTER TABLE ステートメント内で使用されるとき、GENERATED ALWAYS は GENERATED と同等になります。

関連タスク:

- 108 ページの『新しい表に生成列を定義する』
- 111 ページの『新しい表での ID 列の定義』

ID 列定義の変更

手順:

表を再作成した後でインポートまたはロード操作が実行される場合、および表に ID 列がある場合、表の内容を再作成した後で、 ID 値を 1 から生成するようにリセットされます。再作成された表に新規行を挿入する際に、 ID 列を再び 1 から始めたり、 ID 列に重複値を含めたりしないで済むようにするには、以下のようにしてください。

1. 表を再作成します。

2. MODIFIED BY IDENTITYOVERRIDE 文節を使用してデータを表にロードします。データは表にロードされますが、行の識別値は生成されません。
3. 次の照会を実行して、ID 列の最新のカウンター値を入手します。

   ```
   SELECT MAX(<IDENTITY column>)
   ```

   これで、表の ID 列値であったものと等しい値が戻されます。
4. ALTER TABLE ステートメントの RESTART 文節を使用します。

   ```
   ALTER TABLE <table name> ALTER COLUMN <IDENTITY column>
      RESTART WITH <last counter value>
   ```
5. 新規行を表に挿入します。 RESTART WITH 文節で指定された値に基づいて、ID 列値が生成されます。

関連資料:

- 「*SQL リファレンス 第 1 巻*」の『MAX 集約関数』
- 「*SQL リファレンス 第 2 巻*」の『ALTER TABLE ステートメント』
- 「コマンド・リファレンス」の『LOAD コマンド』

制約の変更

制約は、それらをドロップした後、代わりに新しいものを追加することによってのみ変更できます。詳細については、以下の部分を参照してください。

- 『制約の追加』
- 194 ページの『ユニーク制約のドロップ』

制約の追加

ALTER TABLE ステートメントを使用して制約を追加します。このステートメント (構文を含む) の詳細については、「*SQL リファレンス*」を参照してください。

ユニーク制約の追加

手順:

ユニーク制約を既存の表に追加することができます。制約名は、ALTER TABLE ステートメント内に指定した他のどの制約とも同じにすることはできず、その表内でユニークなものでなければなりません (これには、定義されたすべての参照保全制約の名前も含まれます)。ステートメントの完了前に、既存のデータが新しい条件に照らしてチェックされます。

以下の SQL ステートメントは、表内の従業員をユニークなものとして識別するための新しい方法を表す、 EMPLOYEE 表に対するユニーク制約を追加します。

```
ALTER TABLE EMPLOYEE
   ADD CONSTRAINT NEWID UNIQUE(EMPNO,HIREDATE)
```

関連タスク:

- 102 ページの『ユニーク制約の定義』
- 194 ページの『ユニーク制約のドロップ』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『ALTER TABLE ステートメント』

主キーの追加

手順:

大きな表に制約を追加する場合は、まず表をチェック・ペンディング状態にし、制約を追加してから、表をチェックして違反行の統合リストを求めるのが効率的です。明示的にチェック・ペンディング状態を設定するには、SET INTEGRITY ステートメントを使います。表が親表の場合は、すべての従属表および子孫表に対しても暗黙のうちにチェック・ペンディングが設定されます。

コントロール・センターを使用して主キーを追加するには、以下のようにします。

1. オブジェクト・ツリーを順に展開し、**「表 (Tables)」**フォルダーを表示します。
2. 修正したい表を右クリックし、ポップアップ・メニューから**「更新 (Alter)」**を選択します。
3. **「主キー (Primary Key)」**ページで、 1 つまたは複数の列を主キーとして選択し、矢印をクリックして移動します。
4. オプション: 主キーの制約名を入力します。
5. **「OK」**をクリックします。

コマンド行を使用して主キーを追加するには、以下のように入力します。

```
ALTER TABLE <name>
   ADD CONSTRAINT <column_name>
   PRIMARY KEY <column_name>
```

関連タスク:

- 192 ページの『外部キーの追加』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『ALTER TABLE ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『SET INTEGRITY ステートメント』

外部キーの追加

手順:

外部キーが表に追加される場合、以下のステートメントが含まれるパッケージまたはキャッシュに入った動的 SQL は、無効のマークが付けられます。

- 外部キーが入っている表の挿入または更新を行うステートメント
- 親表の更新または削除を行うステートメント

コントロール・センターを使用して外部キーを追加するには、以下のようにします。

1. オブジェクト・ツリーを順に展開し、**「表 (Tables)」**フォルダーを表示します。
2. 修正したい表を右クリックし、ポップアップ・メニューから**「更新 (Alter)」**を選択します。
3. **「外部キー (Foreign Keys)」**ページで、**「追加 (Add)」**をクリックします。
4. **「外部キーの追加 (Add Foreign Keys)」**ウィンドウで、親表の情報を指定します。
5. 1 つまたは複数の列を外部キーとして選択し、矢印をクリックして移動します。
6. 親表の行が削除されたり更新されたりした場合に従属表に対して行うアクションを指定します。外部キーの制約名を追加することもできます。
7. **「OK」**をクリックします。

コマンド行を使用して外部キーを追加するには、以下のように入力します。

```
ALTER TABLE <name>
   ADD CONSTRAINT <column_name>
    FOREIGN KEY <column_name>
    ON DELETE <action_type>
    ON UPDATE <action_type>
```

以下の例は、表に主キーと外部キーを追加するための ALTER TABLE ステートメントを示しています。

```
ALTER TABLE PROJECT
  ADD CONSTRAINT PROJECT_KEY
      PRIMARY KEY (PROJNO)
ALTER TABLE EMP_ACT
  ADD CONSTRAINT ACTIVITY_KEY
      PRIMARY KEY (EMPNO, PROJNO, ACTNO)
  ADD CONSTRAINT ACT_EMP_REF
      FOREIGN KEY (EMPNO)
      REFERENCES EMPLOYEE
      ON DELETE  RESTRICT
  ADD CONSTRAINT ACT_PROJ_REF
      FOREIGN KEY (PROJNO)
      REFERENCES PROJECT
      ON DELETE CASCADE
```

関連概念:

- 219 ページの『オブジェクトを変更するときのステートメント従属関係』

関連タスク:

- 192 ページの『主キーの追加』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『ALTER TABLE ステートメント』

表チェック制約の追加

手順:

チェック制約は、ALTER TABLE ステートメントを使用して既存の表に追加することができます。制約名は、ALTER TABLE ステートメント内に指定した他のどの制約とも同じにすることはできず、その表内でユニークなものでなければなりません (これには、定義されたすべての参照保全制約の名前も含まれます)。ステートメントの完了前に、既存のデータが新しい条件に照らしてチェックされます。

大きな表に制約を追加するには、表をチェック・ペンディング状態にし、制約を追加してから、表をチェックして制約に違反する行の統合リストを求めるのが効率的です。明示してチェック・ペンディング状態を設定するには、SET INTEGRITY ステートメントを使用します。表が親表である場合、チェック・ペンディングは、すべての従属表と子孫表にも暗黙に設定されます。

表チェック制約が追加された場合、表の挿入または更新を行うパッケージおよびキャッシュに入った動的 SQL は、無効のマークが付けられます。

コントロール・センターを使用して表チェック制約を追加するには、以下のようにします。

1. オブジェクト・ツリーを順に展開し、**「表 (Tables)」**フォルダーを表示します。
2. 修正したい表を右クリックし、ポップアップ・メニューから**「更新 (Alter)」**を選択します。
3. **「チェック制約 (Check Constraints)」**ページで、**「追加 (Add)」**をクリックします。
4. **「チェック制約の追加 (Add Check Constraint)」**ウィンドウで、情報をすべて入力して、**「OK」**をクリックします。
5. **「チェック制約 (Check Constraints)」**ページで、**「OK」**をクリックします。

コマンド行を使用して表チェック制約を追加するには、以下のように入力します。

```
ALTER TABLE <name>
   ADD CONSTRAINT <name> (<constraint>)
```

次の SQL ステートメントは、各従業員の給与と歩合給の合計が $25,000 を超えなければならないという制約を EMPLOYEE 表に追加するものです。

```
ALTER TABLE EMPLOYEE
   ADD CONSTRAINT REVENUE CHECK (SALARY + COMM > 25000)
```

関連概念:

- 219 ページの『オブジェクトを変更するときのステートメント従属関係』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『ALTER TABLE ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『SET INTEGRITY ステートメント』

ユニーク制約のドロップ

ALTER TABLE ステートメントを使用して制約をドロップします。このステートメント (構文を含む) の詳細については、「*SQL リファレンス*」を参照してください。

ユニーク制約のドロップ

手順:

ALTER TABLE ステートメントを使用して、明示的にユニーク制約をドロップすることができます。表上のすべてのユニーク制約の名前は、SYSCAT.INDEXES システム・カタログ・ビューの中にあります。

以下の SQL ステートメントは、EMPLOYEE 表からユニーク制約 NEWID をドロップします。

```
ALTER TABLE EMPLOYEE
   DROP UNIQUE NEWID
```

このユニーク制約をドロップすると、その制約を使用しているすべてのパッケージまたはキャッシュに入った動的 SQL を無効にします。

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『ALTER TABLE ステートメント』

主キーのドロップ

手順:

コントロール・センターを使用して主キーをドロップするには、以下のようにします。

1. オブジェクト・ツリーを順に展開し、**「表 (Tables)」**フォルダーを表示します。
2. 修正したい表を右クリックし、ポップアップ・メニューから**「更新 (Alter)」**を選択します。
3. **「主キー (Primary Key)」**ページの右側で、ドロップする主キーを選択し、矢印をクリックして左側の**「使用可能な列 (Available columns)」**ボックスに移動します。
4. **「OK」**をクリックします。

コマンド行を使用して主キーをドロップするには、以下のように入力します。

```
ALTER TABLE <name>
  DROP PRIMARY KEY
```

外部キーがドロップされると、以下のものを含むパッケージまたはキャッシュに入った動的 SQL ステートメントは、無効のマークが付けられます。

- 外部キーが入っている表の挿入または更新を行うステートメント
- 親表の更新または削除を行うステートメント

関連概念:

- 219 ページの『オブジェクトを変更するときのステートメント従属関係』

関連タスク:

- 195 ページの『外部キーのドロップ』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『ALTER TABLE ステートメント』

外部キーのドロップ

手順:

コントロール・センターを使用して外部キーをドロップするには、以下のようにします。

1. オブジェクト・ツリーを順に展開し、**「表 (Tables)」**フォルダーを表示します。
2. 修正したい表を右クリックし、ポップアップ・メニューから**「更新 (Alter)」**を選択します。
3. **「外部キー (Foreign Keys)」**ページで、**「追加 (Add)」**をクリックします。
4. 右側でドロップする外部キーを選択し、矢印をクリックして左側の**「使用可能な列 (Available columns)」**ボックスに移動します。
5. **「外部キー (Foreign Keys)」**ページで、**「OK」**をクリックします。

コマンド行を使用して外部キーをドロップするには、以下のように入力します。

```
ALTER TABLE <name>
   DROP FOREIGN KEY <foreign_key_name>
```

以下の例は、 ALTER TABLE ステートメントの DROP PRIMARY KEY および DROP FOREIGN KEY 文節を使用して、表の主キーと外部キーをドロップします。

```
ALTER TABLE EMP_ACT
  DROP PRIMARY KEY
  DROP FOREIGN KEY ACT_EMP_REF
  DROP FOREIGN KEY ACT_PROJ_REF
ALTER TABLE PROJECT
  DROP PRIMARY KEY
```

関連概念:

- 219 ページの『オブジェクトを変更するときのステートメント従属関係』

関連タスク:

- 195 ページの『主キーのドロップ』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『ALTER TABLE ステートメント』

表チェック制約のドロップ

手順:

表チェック制約を明示的にドロップまたは変更するには、ALTER TABLE ステートメントを使います。また、DROP TABLE ステートメントを使うと、暗黙のうちにチェック制約がドロップされます。

表チェック制約をドロップすると、その表上で INSERT または UPDATE 依存関係を持つすべてのパッケージおよびキャッシュに入った動的 SQL ステートメントは、無効にされます。表内のすべてのチェック制約の名前は、SYSCAT.CHECKS カタログ・ビューの中にあります。名前がシステム生成である表チェック制約のドロップを試行する前に、 SYSCAT.CHECKS カタログ・ビューでその名前を探してください。

コントロール・センターを使用して表チェック制約をドロップするには、以下のようにします。

1. オブジェクト・ツリーを順に展開し、**「表 (Tables)」**フォルダーを表示します。
2. 修正したい表を右クリックし、ポップアップ・メニューから**「更新 (Alter)」**を選択します。
3. **「チェック制約 (Check Constraints)」**ページで、ドロップするチェック制約を選択し、**「ドロップ (Remove)」**をクリックして、**「OK」**をクリックします。

コマンド行を使用して表チェック制約をドロップするには、以下のようにします。

```
ALTER TABLE <table_name>
   DROP CHECK <check_constraint_name>
```

次の SQL ステートメントは、 EMPLOYEE 表から REVENUE という表チェック制約をドロップするものです。

```
ALTER TABLE EMPLOYEE
   DROP CHECK REVENUE
```

関連概念:

• 219 ページの『オブジェクトを変更するときのステートメント従属関係』

関連タスク:

• 193 ページの『表チェック制約の追加』

関連資料:

• 「*SQL リファレンス 第 2 巻*」の『ALTER TABLE ステートメント』

既存の表での生成列の定義

生成列は基本表に定義されます。生成列に格納される値は、式を使って計算された値です。挿入操作や更新操作で指定された値ではありません。生成列は、表を作成する時、または既存の表を修正する時に作成できます。

前提条件:

生成列を定義できるのは、等価性の比較が定義されているデータ・タイプだけです。生成列を定義できないデータ・タイプとしては、構造型、 LOB、 CLOB、 DBCLOB、 LONG VARCHAR、 LONG VARGRAPHIC、およびこれらのデータ・タイプを使用して定義したユーザー定義のタイプがあります。

制約、ユニーク索引、参照制約、主キー、およびグローバル一時表には生成列を使用できません。 LIKE を使用して作成した表と、マテリアライズしたビューは、生成列の特性を継承しません。

制約事項:

キーワード DEFAULT を指定しないと、生成列を挿入したり更新したりできません。挿入の際に DEFAULT を使用すると、列リストに列を列挙する必要がなくなります。代わりに値リストの中で生成列を DEFAULT に設定できます。更新の際に DEFAULT を使用すると、 SET INTEGRITY によってチェック抜きでオンラインで配置された生成列を再計算できます。

トリガー処理の順序の関係上、BEFORE トリガーのヘッダー (更新前) または本体が、生成列を参照するようにしないでください。処理の順序としては、生成列は BEFORE トリガーの後に処理されます。

db2look ユーティリティーは、生成列によって生成されるチェック制約を考慮に入れません。

複製を使用する際には、ターゲット表のマッピングに生成列を使用してはなりません。複製する際には 2 つの選択肢があります。

- ターゲット表に、正規の (つまり、生成列ではない) 列として生成列が定義されていなければならない。
- ターゲット表のマッピングから生成列を省略しなければならない。

生成列を処理する際には複数の制約事項があります。

- 生成列同士は従属関係にあってはならない。
- 生成列の作成に使用する式に、副照会を含めることはできない。これには、READS SQL DATA を実行する関数を使用する式も含まれます。
- 生成列にはチェック制約は使用できない。

手順:

生成列を定義するには、以下のステップを実行します。

1. 表をチェック・ペンディング状態にします。

   ```
   SET INTEGRITY FOR t1 OFF
   ```

2. 表を変更して、1 つまたは複数の生成列を追加します。

   ```
   ALTER TABLE t1 ADD COLUMN c3 DOUBLE GENERATED ALWAYS AS (c1 + c2),
                  ADD COLUMN c4 GENERATED ALWAYS AS
                     (CASE WHEN c1 > c3 THEN 1 ELSE NULL END))
   ```

3. 生成列に正確な値を割り当てます。このことは、以下の方式を用いて行います。
   - 次のようにして、生成列の値を再計算して再割り当てします。

     ```
     SET INTEGRITY FOR t1 IMMEDIATE CHECKED FORCE GENERATED
     ```

     その SET INTEGRITY ステートメントがログ・スペースの不足のために失敗する場合には、使用可能なアクティブ・ログ・スペースを増やして、再度 SET INTEGRITY ステートメントを実行します。

     注: この時点で例外表を使用できます。
   - 使用可能なアクティブ・ログ・スペースを増やすことができない場合には、検索済み更新ステートメントを使用して、生成列をデフォルト値に割り当ててください。
     a. 表の排他ロックを取得します。そうすると、非コミット読み取り以外のトランザクションは表にアクセスできなくなります。表ロックは最初の断続コミットの時に解除され、他のトランザクションが、まだデフォルト値に割り当てられていない生成列によって行を見ることができてしまうという点に注意してください。

        ```
        LOCK TABLE t1
        ```

     b. 生成列のチェックをバイパスします。

```
SET INTEGRITY FOR t1 GENERATED COLUMN IMMEDIATE UNCHECKED
```

c. 表の他の保全性違反について (該当する場合) チェックし、そのチェック・ペンディングを解除します。

```
SET INTEGRITY FOR t1 IMMEDIATE CHECKED
```

d. ログが満杯にならないよう、非再現性コミットと述部を使用して生成列を更新します。

```
UPDATE t1 SET (c3, c4) = (DEFAULT, DEFAULT) WHERE <predicate>
```

e. コミット・ステートメントを使用してトランザクションを完了し、表をアンロックします。

```
COMMIT
```

- 使用可能なアクティブ・ログ・スペースを増やすことができない場合には、カーソルを基にしたアプローチも用いられることがあります。

a. 表に FOR UPDATE カーソルを宣言します。断続コミットの後ロックを保存する必要がある場合には、WITH HOLD オプションを使用する必要があります。

```
DECLARE C1 CURSOR WITH HOLD FOR S1
```

ここで S1 は次のように定義されます。

```
SELECT '0' FROM t1 FOR UPDATE OF C3, C4
```

b. カーソルをオープンします。

```
OPEN C1
```

c. 生成列のチェックをバイパスします。

```
SET INTEGRITY FOR t1 GENERATED COLUMN IMMEDIATE UNCHECKED
```

d. 表の他の保全性違反について (該当する場合) チェックし、そのチェック・ペンディングを解除します。

```
SET INTEGRITY FOR t1 IMMEDIATE CHECKED
```

e. 表のすべての行を取り出すためにループを持ち、取り出した行ごとに以下を実行して、生成列をそのデフォルト表に割り当てます。カーソルの継続時間中に表が確実にロックされているようにするため、最初のフェッチは、必ず表がチェック・ペンディングから解除された直後に行うようにすることが大切です。

```
UPDATE t1 SET (C3, C4) = (DEFAULT, DEFAULT) WHERE CURRENT OF C1
```

断続コミットを行い、ログが満杯になることを避けてください。

f. カーソルをクローズして、表をアンロックするためにコミットします。

```
CLOSE C1
COMMIT
```

- NOT LOGGED INITIALLY オプションを指定して表が作成されたことが分かっている場合。この場合、表に関するロギングは無効になっています。これには、生成列値の処理中に、通常の影響やリスクが伴います。

a. NOT LOGGED INITIALLY オプションをアクティブにします。

```
ALTER TABLE t1 ACTIVATE NOT LOGGED INITIALLY
```

b. 値を生成します。

```
SET INTEGRITY FOR t1 IMMEDIATE CHECKED FORCE GENERATION
```

c. トランザクションをコミットして、 NOT LOGGED INITIALLY オプションを再度無効にします。

```
COMMIT
```

等価性のチェック制約のように、式を適用することによって、生成列の値に単純なチェックを加えることもできます。

```
SET INTEGRITY FOR t1 IMMEDIATE CHECKED
```

生成列に (たとえば LOAD を使用して) 値を入れ、かつその値が、生成された式に一致することが分かっている場合、値のチェックまたは割り当てを行わずに、表をチェック・ペンディング状態から解除することができます。

```
SET INTEGRITY FOR t1 GENERATED COLUMN IMMEDIATE UNCHECKED
```

関連タスク:

- 108 ページの『新しい表に生成列を定義する』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『ALTER TABLE ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『COMMIT ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『LOCK TABLE ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『SET INTEGRITY ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『UPDATE ステートメント』
- 「コマンド・リファレンス」の『db2look - DB2 統計および DDL 抽出ツール・コマンド』

表の揮発性を宣言する

手順:

揮発性 表は、ランタイムにその内容が空であるものから内容が非常に大きなものに至るまで、さまざまな内容を持つ表として定義されます。この種の表には揮発性、つまり極端な変更の可能性があるため、 RUNSTATS が収集する統計の信頼性が不確かなものになります。 統計は単一ポイント・イン・タイムで収集され、表面的なものにすぎません。揮発性表を使用するアクセス・プランを生成しても、実行プランが不正確または貧弱なものになる可能性があります。 たとえば、揮発性表が空のときに統計を収集すると、オプティマイザーの傾向として、索引スキャンではなく表スキャンを使って揮発性表にアクセスするほうが優先されます。

この傾向を回避するには、ALTER TABLE ステートメントを使って、表を揮発性として宣言することを考慮してください。表を揮発性として宣言すれば、オプティマイザーは表スキャンではなく索引スキャンの使用を考慮します。宣言された揮発性表を使用するアクセス・プランは、該当する表の既存統計には依存しません。

コントロール・センターを使用して表の揮発性を宣言するには、以下のようにします。

1. オブジェクト・ツリーを順に展開し、**「表 (Tables)」**フォルダーを表示します。
2. 修正したい表を右クリックし、ポップアップ・メニューから**「更新 (Alter)」**を選択します。
3. **「表 (Table)」**ページで、**「カーディナリティーはランタイムに変化します (Cardinality varies significantly at run time)」**チェック・ボックスを選択し、**「OK」**をクリックします。

コマンド行を使用して、表を 『揮発性』 として宣言するには、以下のように入力します。

```
ALTER TABLE <table_name>
   VOLATILE CARDINALITY
```

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『ALTER TABLE ステートメント』

パーティション・キーの変更

手順:

単一パーティションのデータベース・パーティション・グループ内の表にあるパーティション・キーのみを変更することができます。まず既存のパーティション・キーをドロップしてから、別のパーティション・キーを作成してください。

以下の SQL ステートメントは、MIXREC 表からパーティション・キー MIX_INT をドロップします。

```
ALTER TABLE MIXREC
   DROP PARTITIONING KEY
```

複数パーティションのデータベース・パーティション・グループの中の表にあるパーティション・キーは、変更できません。このパーティション・キーをドロップしようとすると、エラーが戻されます。

複数パーティションのデータベース・パーティション・グループのパーティション・キーを変更するには、次のどちらかを行ってください。

- すべてのデータを単一パーティションのデータベース・パーティション・グループにエクスポートしてから、上記の指示に従う。
- すべてのデータをエクスポートして、表をドロップし、パーティション・キーを再定義して表を再作成してから、すべてのデータをインポートする。

これらの方法はいずれも、大きなデータベースの場合は現実的ではありません。このため、大きなデータベースの設計を実現する前に、適切なパーティション・キーを定義しておくことが最も重要です。

関連概念:

- 「*管理ガイド: プランニング*」の『パーティション・キー』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『ALTER TABLE ステートメント』

表属性の変更

手順:

データ・キャプチャー・オプション、各ページでのフリー・スペースのパーセンテージ (PCTFREE)、ロック・サイズ、または追加モードなどの表属性を変更する必要があるかもしれません。

表の各ページに残すフリー・スペースの量は、PCTFREE によって指定しますが、これはクラスター索引を効果的に使用するための重要な考慮事項です。指定する量は、既存のデータと予想される将来のデータの性質によって決まります。 PCTFREE の指定は、LOAD と REORG については有効ですが、挿入、更新、およびインポート活動では無視されます。

PCTFREE を大きな値に設定すると、クラスタリングを長い期間保つことができますが、ディスク・スペースもより多く必要となります。

LOCKSIZE パラメーターを使用して、表がアクセスされるときに使用されるロックのサイズ (細分性) を指定できます。表が作成されるとき、デフォルトでは行レベルのロックが定義されます。表レベルのロックを使用すると、獲得し解放する必要のあるロックの数を限定することによって、照会のパフォーマンスが向上します。

APPEND ON を指定すると、全体の表のパフォーマンスを向上させることができます。これによって、より速い挿入が可能となり、一方ではフリー・スペースについての情報の保守が不要になります。

クラスタリング索引のある表は、追加モードをオンにするように変更することはできません。同様に、追加モードの表でクラスタリング索引を作成することはできません。

関連概念:

- 「*管理ガイド: パフォーマンス*」の『ロックおよび並行性の制御』
- 「*管理ガイド: パフォーマンス*」の『ロック属性』
- 「*管理ガイド: パフォーマンス*」の『ロックとパフォーマンス』
- 「*管理ガイド: パフォーマンス*」の『ロッキングに影響を与える要因』
- 「*管理ガイド: パフォーマンス*」の『ロッキングのガイドライン』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『ALTER TABLE ステートメント』

ID 列の変更

手順:

ALTER TABLE ステートメントで、既存の ID 列の属性を変更します。

シーケンスの特性を持つように ID 列を変更するためには、いくつかの方法があります。

ALTER TABLE および ID 列には、以下のようなユニークなタスクがあります。

- RESTART は、ID 列に関連付けられたシーケンスを、ID 列が最初に作成されたときに明示的または暗黙的に指定された値にリセットします。
- RESTART WITH <numeric-constant> は、ID 列に関連付けられたシーケンスを数値定数にリセットします。数値定数は、ID 列に割り当てることができる、小数点以下に非ゼロ桁がない正または負の値です。

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『ALTER TABLE ステートメント』

シーケンスの変更

手順:

ALTER SEQUENCE ステートメントで、既存のシーケンスの属性を変更します。

変更可能なシーケンスの属性

- 今後の値の間の増分を変更
- 新しい最小値または最大値を確立
- キャッシュ済みシーケンス番号の数を変更
- シーケンスが循環するかどうかを変更
- 要求の順序でシーケンス番号が生成されるかどうかを変更
- シーケンスを再始動

シーケンス作成の一部ではないタスクが 2 つあります。追加される値は以下の通りです。

- RESTART。シーケンスを、そのシーケンスの作成時に開始値として明示的または暗黙的に指定された値にリセットします。
- RESTART WITH <numeric-constant>。シーケンスを数値定数にリセットします。数値定数は、小数点以下に非ゼロ桁がない正または負の値です。

シーケンスを再始動、または CYCLE に変更した後、重複するシーケンス番号が生成される可能性があります。今後のシーケンス番号だけが ALTER SEQUENCE ステートメントによって影響を受けます。

シーケンスのデータ・タイプは変更できません。その代わり、現行シーケンスをドロップして、新しいデータ・タイプを指定した新しいシーケンスを作成する必要があります。

シーケンスが変更されると、DB2 で使用されていないキャッシュ済みの値はすべて失われます。

関連タスク:

- 204 ページの『シーケンスのドロップ』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『ALTER SEQUENCE ステートメント』

シーケンスのドロップ

手順:

シーケンスを削除するには、DROP ステートメントを使用します。

以下を実行すると、特定のシーケンスをドロップすることができます。

```
DROP SEQUENCE sequence_name
```

sequence_name はドロップするシーケンス名で、ここには、既存のシーケンスを正しく識別するための暗黙的または明示的なスキーマ名が入ります。

ID 列のためにシステム作成されたシーケンスは、DROP SEQUENCE ステートメントを使用してドロップすることはできません。

シーケンスをドロップすると、そのシーケンスの特権もすべてドロップされます。

関連タスク:

- 203 ページの『シーケンスの変更』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『DROP ステートメント』

マテリアライズ照会表のプロパティーの変更

手順:

多少の制約事項はありますが、マテリアライズ照会表を正規表に変更したり、正規表をマテリアライズ照会表に変更したりできます。他の表タイプは変更できません。変更できるのは正規表とマテリアライズ照会表だけです。たとえば、複製マテリアライズ照会表から正規表に (あるいはその逆に) 変更することはできません。

正規表をマテリアライズ照会表に変更すると、その表はチェック・ペンディング状態になります。このように変更した場合、マテリアライズ照会表の定義を全選択したものは変更前の表の定義と一致していなければなりません。つまり、以下の点を満たしていなければなりません。

- 列の数が同じでなければならない。
- 列名と位置が一致しなければならない。
- データ・タイプが同一でなければならない。

マテリアライズ照会表が元の表で定義されている場合、元の表自体を変更してマテリアライズ照会表にすることはできません。元の表にトリガー、チェック制約、参照制約、または定義済みのユニーク索引がある場合、その表をマテリアライズ照会表に変更することはできません。マテリアライズ照会表を定義するために表の特性を変更する場合、同じ ALTER TABLE ステートメントを使って、別の方法で表を変更することはできません。

正規表をマテリアライズ照会表に変更する場合、マテリアライズ照会表の定義の全選択では、元の表を直接参照したり、あるいはビュー、別名、またはマテリアライズ照会表を介して間接的に参照することはできません。

マテリアライズ照会表を正規表に変更するには、以下のようにします。

```
ALTER TABLE sumtable
   SET SUMMARY AS DEFINITION ONLY
```

正規表をマテリアライズ照会表に変更するには、以下のようにします。

```
ALTER TABLE regtable
   SET SUMMARY AS <fullselect>
```

正規表をマテリアライズ照会表に変更する際の全選択に関する制約事項は、CREATE SUMMARY TABLE ステートメントを使用してサマリー表を作成する際の制約事項によく似ています。

関連タスク:

- 136 ページの『マテリアライズ照会表の作成』
- 205 ページの『マテリアライズ照会表のデータのリフレッシュ』
- 216 ページの『マテリアライズ照会またはステージング表のドロップ』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『ALTER TABLE ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『CREATE TABLE ステートメント』

マテリアライズ照会表のデータのリフレッシュ

手順:

REFRESH TABLE ステートメントを使用して、 1 つ以上のマテリアライズ照会表のデータをリフレッシュできます。このステートメントは、アプリケーション・プログラムに組み込むこともできますし、動的に出すこともできます。このステートメントを使用するには、SYSADM または DBADM 権限か、更新する表に対する CONTROL 特権が必要です。

次の例は、マテリアライズ照会表のデータをリフレッシュする方法を示しています。

```
REFRESH TABLE SUMTAB1
```

関連タスク:

- 136 ページの『マテリアライズ照会表の作成』
- 204 ページの『マテリアライズ照会表のプロパティーの変更』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『REFRESH TABLE ステートメント』

ユーザー定義構造型タイプの変更

手順:

構造型の作成後に、その構造型に関連した属性を追加またはドロップしなければならないことに気付くことがあるでしょう。これは、ALTER TYPE (構造化) ステートメントを使用して行われます。

関連概念:

- 「*アプリケーション開発ガイド サーバー・アプリケーションのプログラミング*」の『ユーザー定義構造化型』
- 「*アプリケーション開発ガイド サーバー・アプリケーションのプログラミング*」の『構造化型階層』

関連タスク:

- 「*アプリケーション開発ガイド サーバー・アプリケーションのプログラミング*」の『構造化型の作成』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『ALTER TYPE (構造化) ステートメント』

型付き表の行の削除および更新

検索済み DELETE ステートメントか、位置決め済み DELETE ステートメントのいずれかを使用して、行を型付き表から削除できます。検索済み UPDATE ステートメントか、位置決め済み UPDATE ステートメントのいずれかを使用して、型付き表の行を更新できます。

関連概念:

- 「*アプリケーション開発ガイド サーバー・アプリケーションのプログラミング*」の『型付き表』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『DELETE ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『UPDATE ステートメント』

既存の表または索引の名前変更

スキーマ内の既存の表または索引に新しい名前を与え、オリジナルの表に作成されていた許可と索引はそのまま維持することができます。

前提条件:

名前変更する既存の表または索引は、表または索引を識別する別名であっても構いません。

制約事項:

名前変更する既存の表または索引は、カタログ表または索引、サマリー表または索引、型付き表、宣言されているグローバル一時表、ニックネーム、または表やビューや別名以外のオブジェクトの名前であってはなりません。

既存の表または索引は、以下のいずれでも参照できません。

- ビュー
- トリガー
- 参照制約

- サマリー表
- 既存の参照列の有効範囲

また、表内にチェック制約があったり、 ID 列以外の生成列があったりしてはなりません。オリジナルの表に依存するパッケージまたはキャッシュに入った動的 SQL ステートメントは、すべて無効にされます。オリジナルの表を参照している別名は修正されません。

名前変更している表または索引が、これらの制約事項のいずれの影響も受けないようにするために、該当のシステム・カタログ表を検査することを考慮する必要があります。

手順:

コントロール・センターを使用して既存の表または索引の名前を変更するには、以下のようにします。

1. オブジェクト・ツリーを順に展開し、**「表 (Tables)」**または**「ビュー (Views)」**フォルダーを表示します。
2. 名前変更したい表またはビューを右クリックし、ポップアップ・メニューから**「名前変更 (Rename)」**を選択します。
3. 新しい表名またはビュー名を入力して、**「OK」**をクリックします。

コマンド行を使用して既存の表の名前を変更するには、以下のように入力します。

```
RENAME TABLE <schema_name>.<table_name> TO <new_name>
```

下記の SQL ステートメントは、 COMPANY スキーマ内の EMPLOYEE 表の名前を EMPL に変更します。

```
RENAME TABLE COMPANY.EMPLOYEE TO EMPL
```

コマンド行を使用して既存の索引の名前を変更するには、以下のように入力します。

```
RENAME INDEX <schema_name>.<index_name> TO <new_name>
```

下記の SQL ステートメントは、 COMPANY スキーマ内の EMPIND 索引の名前を MSTRIND に変更します。

```
RENAME INDEX COMPANY.EMPIND TO MSTRIND
```

パッケージは無効にされるので、名前変更されたばかりの表または索引を参照する場合は、そのパッケージを再バインドしなければなりません。パッケージは、同じ名前を持つ別の索引が存在するかどうかにかかわらず、暗黙的に再バインドされます。他に良い選択肢がなければ、パッケージは以前と同じ索引を新規の名前で使用します。

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『RENAME ステートメント』

MERGE ステートメントを使用して、表およびビューの内容を更新する

DB2 Universal Database では、別のソースからのデータ (通常は、表参照の結果) を使用して表またはビューを更新する機能を提供します。このタイプの更新は、MERGE ステートメントを使用して実行されます。

MERGE ステートメントからの指定された指示に基づいて、ソースと一致するターゲット表内の行を削除または更新することができます。また、ターゲット表内に存在しない行を挿入することができます。

ビュー内の行を更新、削除、または挿入すると、そのビューがベースとなっている表内の対応する行の更新、削除、または挿入が行われます。

制約事項:

MERGE ステートメントに関連する許可 ID は、 3 つの考えられるアクション (ビュー内の表または基礎表での更新、削除、または挿入) のいずれかを実行する適切な特権を持っている必要があります。許可 ID は、副照会内のビュー内の表または基礎表にも適切な特権を持っている必要があります。

MERGE ステートメントでエラーが発生する場合、 MERGE に関連した操作全体がロールバックされます。

MERGE ステートメントが実行される前に存在しなかった、ビューのターゲット表または基礎表内の行を更新することはできません。つまり、MERGE ステートメントの一部として挿入された行の更新は許可されません。

MERGE ステートメントのターゲットとしてビューが指定される場合、そのビューに対して INSTEAD OF トリガーが定義されてはなりません。または、それぞれの更新、削除、挿入操作に関して INSTEAD OF トリガーが定義されなければなりません。

手順:

ターゲット表でこれらのアクションの組み合わせを更新、削除、挿入、または実行するには、コマンド・プロンプトで以下のように入力します。

```
MERGE INTO <table or view name>
   USING <table reference> ON <search condition>
   WHEN <match condition> THEN <modification operation or signal statement>
```

変更操作および SIGNAL ステートメントは、 MERGE ステートメントごとに何度も指定することができます。ターゲット表またはビュー内のそれぞれの行は、単一の MERGE ステートメント内で 1 度だけ操作することができます。これは、ターゲット表またはビュー内の 1 行を、表参照の結果表内の 1 行とだけ MATCHED (一致) するとして識別することができることを意味しています。

出荷および在庫という 2 つの表がある状態を考えてみます。出荷表を使用して、行を在庫表にマージします。一致する行の場合、在庫表内の数量ずつ出荷表内の数量を増やします。その他の場合、新しいパーツ番号を在庫表に挿入します。

```
MERGE INTO inventory AS in
   USING (SELECT partno, description, count FROM shipment
   WHERE shipment. partno IS NOT NULL) AS sh
   ON (in.partno = sh.partno)
```

```
WHEN MATCHED THEN
   UPDATE SET
      description = sh.description
      quantity = in.quantity + sh.count
WHEN NOT MATCHED THEN
INSERT
      (partno, description, quantity)
      VALUES (sh.partno, sh.description, sh.count)
```

この例では、DELETE オプションはありません。さらに複雑な一致条件では、DELETE オプションの追加を許可することができます。 SIGNAL ステートメントや ELSE 節の使用など、その他のいくつかのオプションがあります。ここでは示されていませんが、「SQL リファレンス」で扱われています。

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『MERGE ステートメント』

表のドロップ

手順:

表をドロップするには、DROP TABLE SQL ステートメントを使います。

表がドロップされると、その表についての情報が含まれる SYSCAT.TABLES カタログの中の行がドロップされ、その表に依存する他のオブジェクトがあれば、影響を受けます。たとえば、以下のとおりです。

- すべての列名はドロップされます。
- その表の列について作成された索引はドロップされます。
- その表に基づくすべてのビューには作動不能のマークが付けられます。
- ドロップされた表と従属ビューに対するすべての特権が暗黙のうちに取り消されます。
- その表が親表または従属表となっている参照制約がすべてドロップされます。
- ドロップされた表に依存するすべてのパッケージおよびキャッシュに入った動的 SQL ステートメントは、無効のマークが付けられ、従属オブジェクトが再作成されるまで、そのままの状態になります。これには、ドロップされる階層内の副表の上にあるスーパー表に依存しているパッケージが含まれます。
- 参照の有効範囲として、ドロップされた表を定義しているすべての参照列は、「有効範囲解除」されます。
- その表の別名定義は影響を受けません。別名は定義し直すことができません。
- ドロップされた表に依存しているすべてのトリガーには、作動不能のマークが付けられます。
- DATALINK 列によってリンクされているすべてのファイルは、リンク解除されます。リンク解除操作は非同期に実行されますので、それらのファイルは他の操作にすぐに使用することはできません。

コントロール・センターを使用して表をドロップするには、以下のようにします。

1. オブジェクト・ツリーを順に展開し、**「表 (Tables)」**フォルダーを表示します。
2. ドロップしたい表を右クリックして、ポップアップ・メニューから**「ドロップ (Drop)」**を選択します。
3. **「確認 (Confirmation)」**ボックスにチェックを付け、**「OK」**をクリックします。

コマンド行を使用して表をドロップするには、以下のように入力します。

```
DROP TABLE <table_name>
```

次のステートメントは、DEPARTMENT という表をドロップするものです。

```
DROP TABLE DEPARTMENT
```

個々の表に副表がある場合、その表はドロップできません。ただし、次の例に示すとおり、表階層内の表はすべて、単一の DROP TABLE HIERARCHY ステートメントを使ってドロップできます。

```
DROP TABLE HIERARCHY person
```

DROP TABLE HIERARCHY ステートメントでは、ドロップする階層のルート表を指定しなければなりません。

表階層のドロップと特定表のドロップを比較した場合、次のような相違があります。

- DROP TABLE HIERARCHY は、個々の DROP 表ステートメントで活動化される削除トリガーを活動化しない。たとえば、個々の副表をドロップすると、そのスーパー表に対する削除トリガーが活動化されます。
- DROP TABLE HIERARCHY は、ドロップされた表の個々の行についてログ項目を作成しない。代わりに、階層のドロップは単一のイベントとしてログ記録されます。

関連概念:

- 219 ページの『オブジェクトを変更するときのステートメント従属関係』

関連タスク:

- 210 ページの『ユーザー定義の一時表のドロップ』
- 215 ページの『作動不能ビューの回復』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『DROP ステートメント』

ユーザー定義の一時表のドロップ

ユーザー定義一時表は、 DECLARE GLOBAL TEMPORARY TABLE ステートメントを使用して作成されます。

前提条件:

この種の表をドロップする際には、表名がスキーマ名 SESSION で修飾されていなければならず、その表を作成したアプリケーション中になければなりません。

制約事項:

パッケージはこのタイプの表に従属できないので、この種の表をドロップしても無効になりません。

手順:

アクティブな作業単位やセーブポイントより前に作成したユーザー定義の一時表をドロップすると、その表は機能的にドロップされ、アプリケーションはその表にアクセスできなくなります。しかし、その表のスペースは依然として表スペース中に予約されているため、その作業単位がコミットされるかセーブポイントが終了するまで、 USER TEMPORARY 表スペースはドロップできません。

関連タスク:

- 109 ページの『ユーザー定義の一時表の作成』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『DROP ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『SET SCHEMA ステートメント』

トリガーのドロップ

手順:

トリガー・オブジェクトは DROP ステートメントを使用してドロップできますが、この手順によって、以下のように従属パッケージに無効のマークが付けられます。

- 明示列リストなしの更新トリガーがドロップされると、ターゲット表上での更新使用を持つパッケージは無効にされます。
- 列リストつきの更新トリガーがドロップされると、ターゲット表上での更新使用を持つパッケージは、そのパッケージが CREATE TRIGGER ステートメントの column-name リストの中の少なくとも 1 つの列上での更新使用も持っている場合にのみ無効にされます。
- 挿入トリガーがドロップされると、ターゲット表上での挿入使用を持つパッケージが無効にされます。
- 削除トリガーがドロップされると、ターゲット表上での削除使用を持つパッケージが無効にされます。

パッケージは、アプリケーション・プログラムが明示的にバインドまたは再バインドされるか、あるいは、そのパッケージが実行され、データベース・マネージャーが自動的に再バインドを行うまで、無効のままとなります。

関連タスク:

- 123 ページの『トリガーの作成』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『DROP ステートメント』

ユーザー定義関数 (UDF)、関数マッピング、またはメソッドのドロップ

ユーザー定義関数 (UDF)、関数テンプレート、または関数マッピングは、 DROP ステートメントを使ってドロップすることができます。

前提条件:

他のオブジェクトを 1 つの関数または関数テンプレートに従属させることができます。関数をドロップする前に、そのような従属関係 (関数マッピングを含む) をドロップしておかなければなりません。ただし、作動不能のマークが付いているパッケージは例外です。

制約事項:

ある UDF にビュー、トリガー、表チェック制約、または別の UDF が従属している場合、その UDF はドロップできません。 CREATE DISTINCT TYPE ステートメントによって暗黙的に生成された関数はドロップできません。 SYSIBM スキーマまたは SYSFUN スキーマのいずれかの中にある関数は、ドロップできません。

手順:

マッピング・オプション DISABLE を指定すれば、関数マッピングを使用不可にすることができます。

作動不能のマークが付けられたパッケージは、暗黙的には再バインドされません。パッケージは、BIND または REBIND コマンドを使用して再バインドされるか、あるいは、PREP コマンドを使用して準備されなければなりません。 UDF をドロップすると、それを使用していたパッケージまたはキャッシュに入った動的 SQL ステートメントをすべて無効にします。

関数マッピングをドロップすると、パッケージに無効のマークが付けられます。自動再バインドが実行され、オプティマイザーはローカル関数を使用しようとします。ローカル関数がテンプレートである場合、暗黙的な再バインドは失敗します。

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『DROP ステートメント』
- 「コマンド・リファレンス」の『BIND コマンド』
- 「コマンド・リファレンス」の『PRECOMPILE コマンド』
- 「コマンド・リファレンス」の『REBIND コマンド』

ユーザー定義タイプ (UDT) またはタイプ・マッピングのドロップ

ユーザー定義タイプ (UDT) またはタイプ・マッピングは、 DROP ステートメントを使ってドロップすることができます。

制約事項:

以下のように UDT が使用されている場合には UDT をドロップできません。

- 既存の表やビュー (特殊タイプ) の列定義内で

- 既存の型付き表または型付きビューのタイプとして使用されている場合 (構造型)
- 別の構造型のスーパータイプとして

デフォルトのタイプ・マッピングはドロップできません。別のタイプ・マッピングを作成してオーバーライドすることしかできません。

データベース・マネージャーはこの特殊タイプに依存する関数をすべてドロップしようとします。 UDF をドロップできなければ、UDT をドロップできません。ある UDF にビュー、トリガー、表チェック制約、または別の UDF が従属している場合、その UDF はドロップできません。 UDT をドロップすると、それを使用していたパッケージまたはキャッシュに入った動的 SQL ステートメントをすべて無効にします。

管理者または他のアプリケーション開発者によって定義された変換だけをドロップできる点に注意してください。組み込み変換とその関連グループの定義をドロップすることはできません。

手順:

ユーザー定義タイプをドロップするには DROP ステートメントが使用されます。

UDT 用の変換を作成し、UDT のドロップを計画している場合は、変換をドロップする必要があるかどうかを考慮してください。これは、DROP TRANSFORM ステートメントを使って実行されます。

関連概念:

- 130 ページの『ユーザー定義タイプ (UDT)』

関連タスク:

- 131 ページの『ユーザー定義特殊タイプの作成』
- 132 ページの『タイプ・マッピングの作成』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『DROP ステートメント』

ビューの変更またはドロップ

ALTER VIEW ステートメントは、有効範囲を追加するよう参照タイプ列を変更することによって、既存のビュー定義を変更します。 DROP ステートメントはビューを削除します。

前提条件:

ビューを変更するときには、有効範囲がまだ定義されていない既存の参照タイプ列に、有効範囲を追加することが必要です。さらに、列はスーパービューから継承したものであってはなりません。

制約事項:

ビューの基礎となる内容を変更すると、トリガーの使用が必要になります。ビューに加えるその他の変更では、ビューをドロップした後に再作成する必要があります。

手順:

ALTER VIEW ステートメントの列名のデータ・タイプは、 REF (型付き表名または型付きビュー名のタイプ) でなければなりません。 INSTEAD OF トリガーを使用してビューの内容を変更することもできます。

パッケージ、またはキャッシュに入った動的ステートメントに無効のマークが付けられても、表および索引のような他のデータベース・オブジェクトは影響されません。

コントロール・センターを使用してビューの定義を変更するには、以下のようにします。

1. オブジェクト・ツリーを順に展開し、**「ビュー (Views)」**フォルダーを表示します。
2. 修正したいビューを右クリックし、ポップアップ・メニューから**「変更 (Alter)」**を選択します。
3. **「ビューの変更 (Alter view)」**ウィンドウで、注釈を入力するか修正し、**「OK」**をクリックします。

コマンド行を使用してビューを変更するには、以下のように入力します。

```
ALTER VIEW <view_name> ALTER <column name>
   ADD SCOPE <typed table or view name>
```

コントロール・センターを使用してビューをドロップするには、以下のようにします。

1. オブジェクト・ツリーを順に展開し、**「ビュー (Views)」**フォルダーを表示します。
2. ドロップしたいビューを右クリックして、ポップアップ・メニューから**「ドロップ (Drop)」**を選択します。
3. **「確認 (Confirmation)」**ボックスにチェックを付け、**「OK」**をクリックします。

コマンド行を使用してビューをドロップするには、以下のように入力します。

```
DROP VIEW <view_name>
```

以下の例は、EMP_VIEW をドロップする方法を示したものです。

```
DROP VIEW EMP_VIEW
```

ドロップするビューに依存するビューがあれば、それらは作動不能になります。

表階層の場合と同様、階層のルート・ビューを指定しても、次の例に示すとおり、1つのステートメント内でビュー階層全体をドロップすることはできません。

```
DROP VIEW HIERARCHY VPerson
```

関連概念:

- 219 ページの『オブジェクトを変更するときのステートメント従属関係』

関連タスク:

- 123 ページの『トリガーの作成』
- 133 ページの『ビューの作成』
- 215 ページの『作動不能ビューの回復』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『ALTER VIEW ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『DROP ステートメント』

作動不能ビューの回復

手順:

ビューは、次のような場合に*作動不能* になることがあります。

- 基本表での特権が取り消された結果。
- 表、別名、または関数がドロップされる場合。
- スーパービューが作動不能になる場合。 (スーパービューとは、別の型付きビュー (サブビュー) がそれを基にする型付きビューのことです。)
- 従属するビューがドロップされる場合

次のステップは、作動不能ビューを回復するのに役に立ちます。

1. ビューを作成するために最初に使用した SQL ステートメントを判別する。この情報は SYSCAT.VIEW カタログ・ビューの TEXT 列から獲得することができます。
2. CREATE VIEW ステートメントを使用して、同じビュー名と同じ定義でビューを再作成する。
3. GRANT ステートメントを使用して、ビューに以前に付与されていたすべての特権を再度付与する。 (作動不能ビューに付与されていたすべての特権は取り消されていることに注意してください。)

作動不能ビューを回復したくない場合は、 DROP VIEW ステートメントを使用してそのビューを明示的にドロップするか、または同じ名前と別の定義を使用して新規のビューを作成することができます。

作動不能ビューは、SYSCAT.TABLES および SYSCAT.VIEWS カタログ・ビューにしか項目がありません。 SYSCAT.VIEWDEP、SYSCAT.TABAUTH、SYSCAT.COLUMNS、および SYSCAT.COLAUTH カタログ・ビューのすべての項目が除去されます。

関連タスク:

- 213 ページの『ビューの変更またはドロップ』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『CREATE VIEW ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『DROP ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『GRANT (表、ビュー、またはニックネーム特権) ステートメント』

- 「*SQL リファレンス 第 1 巻*」の『SYSCAT.VIEWS カタログ・ビュー』

マテリアライズ照会またはステージング表のドロップ

手順:

マテリアライズ照会表またはステージング表は変更できませんが、ドロップすることはできます。

この表を参照しているすべての索引、主キー、外部キー、およびチェック制約がドロップされます。この表を参照するすべてのビューおよびトリガーは、作動不能になります。ドロップされたオブジェクトまたは作動不能とマークされたオブジェクトに依存するすべてのパッケージは、無効になります。

コントロール・センターを使用してマテリアライズ照会表をドロップするには、以下のようにします。

1. オブジェクト・ツリーを順に展開し、**「表 (Tables)」**フォルダーを表示します。
2. ドロップしたいマテリアライズ照会表またはステージング表を右クリックして、ポップアップ・メニューから**「ドロップ (Drop)」**を選択します。
3. **「確認 (Confirmation)」**ボックスにチェックを付け、**「OK」**をクリックします。

コマンド行を使用してマテリアライズ照会表またはステージング表をドロップするには、以下のように入力します。

```
DROP TABLE <table_name>
```

次の SQL ステートメントは、マテリアライズ照会表 XT をドロップするものです。

```
DROP TABLE XT
```

マテリアライズ照会表は、 DROP TABLE ステートメントを使用して明示的にドロップすることもできますし、基礎表のいずれかがドロップされる場合は暗黙的にドロップすることもできます。

ステージング表は、 DROP TABLE ステートメントを使用して明示的にドロップすることもできますし、関連したマテリアライズ照会表がドロップされるときに暗黙的にドロップすることもできます。

関連概念:

- 219 ページの『オブジェクトを変更するときのステートメント従属関係』

関連タスク:

- 136 ページの『マテリアライズ照会表の作成』
- 142 ページの『ステージング表の作成』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『DROP ステートメント』

作動不能サマリー表の回復

手順:

サマリー表は、基本表での SELECT 特権を取り消されると、*作動不能* になります。

次のステップは、作動不能サマリー表を回復するのに役に立ちます。

- サマリー表を作成するために最初に使用した SQL ステートメントを判別する。この情報は SYSCAT.VIEW カタログ・ビューの TEXT 列から獲得することができます。
- CREATE SUMMARY TABLE ステートメントおよび同じ名前と同じ定義を使用して、サマリー表を再作成する。
- GRANT ステートメントを使用して、サマリー表に以前に付与されていたすべての特権を再度付与する。 (作動不能サマリー表に付与されていたすべての特権は取り消されていることに注意してください。)

作動不能サマリー表を回復したくない場合は、 DROP TABLE ステートメントを使用してそのサマリー表を明示的にドロップするか、または同じ名前と別の定義を使用して新規のサマリー表を作成することができます。

作動不能サマリー表は、SYSCAT.TABLES および SYSCAT.VIEWS カタログ・ビューにしか項目がありません。 SYSCAT.VIEWDEP、SYSCAT.TABAUTH、SYSCAT.COLUMNS、および SYSCAT.COLAUTH カタログ・ビューのすべての項目が除去されます。

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『CREATE TABLE ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『DROP ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『GRANT (表、ビュー、またはニックネーム特権) ステートメント』
- 「*SQL リファレンス 第 1 巻*」の『SYSCAT.VIEWS カタログ・ビュー』

索引、索引の拡張、または索引の指定のドロップ

制約事項:

索引定義、索引の拡張、または索引の指定の文節は変更できません。索引や索引の拡張をドロップしてから再び作成してください。 (索引または索引の指定をドロップしても、他のオブジェクトがドロップされることはありません。しかし、場合によっては一部のパッケージが無効になることがあります。)

索引の拡張の名前は、カタログ中に記述されている索引の拡張を識別するものでなければなりません。 RESTRICT 文節は、索引の拡張の定義に従属する索引は定義できないという規則を施行します。基礎となる索引がこの索引の拡張に従属していると、ドロップは失敗します。

主キーまたはユニーク・キーの索引 (索引の指定でない場合に限る) は、明示的にドロップすることはできません。索引をドロップするには、次の方法のいずれかを使用してください。

- 主キーまたはユニーク・キーに対して、 1 次索引またはユニーク制約が自動的に作成されていた場合、主キーまたはユニーク・キーをドロップすると、索引がドロップされることになります。ドロップは、ALTER TABLE ステートメントによって行われます。
- 1 次索引またはユニーク制約がユーザー定義であった場合、ALTER TABLE を使用して、まず主キーまたはユニーク・キーをドロップしなければなりません。主キーまたはユニーク・キーがドロップされた後は、索引は 1 次索引またはユニーク索引とは考えられなくなるため、明示してドロップすることができます。

手順:

コントロール・センターを使用して索引、索引の拡張、または索引の指定をドロップするには、以下のようにします。

1. オブジェクト・ツリーを順に展開し、**「索引 (Indexes)」**フォルダーを表示します。
2. ドロップしたい索引を右クリックして、ポップアップ・メニューから**「ドロップ (Drop)」**を選択します。
3. **「確認 (Confirmation)」**ボックスにチェックを付け、**「OK」**をクリックします。

コマンド行を使用して索引、索引の拡張、または索引の指定をドロップするには、以下のように入力します。

```
DROP INDEX <index_name>
```

次の SQL ステートメントは、PH という索引をドロップするものです。

```
DROP INDEX PH
```

次の SQL ステートメントは、IX_MAP という索引の拡張をドロップするものです。

```
DROP INDEX EXTENSION ix_map RESTRICT
```

その索引に依存するパッケージ、およびキャッシュに入った動的 SQL ステートメントには、無効のマークが付けられます。アプリケーション・プログラムは、索引の追加やドロップによる変更事項には影響されません。

関連概念:

- 219 ページの『オブジェクトを変更するときのステートメント従属関係』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『ALTER TABLE ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『DROP ステートメント』

オブジェクトを変更するときのステートメント従属関係

ステートメントの従属関係には、パッケージ、およびキャッシュに入った動的 SQL ステートメントが含まれます。パッケージ とは、データベース・オブジェクトの 1 つで、データベース・マネージャーが、特定のアプリケーション・プログラムにとって最も効率的な方法でデータにアクセスするのに必要な情報が入れられたものです。バインド とは、データベース・マネージャーが、アプリケーションの実行時にデータベースにアクセスするのに必要なパッケージを作成するプロセスです。

パッケージおよびキャッシュに入った動的 SQL ステートメントは、さまざまなタイプのオブジェクトに従属することができます。

そうしたオブジェクトは、明示的に参照できます。 SQL SELECT ステートメントに含める表やユーザー定義関数などはその例です。また、暗示的に参照できるオブジェクトもあります。たとえば、親表の行の削除時に、参照制約の違反がないかどうかの検査が必要な従属表がそうです。パッケージはさらに、パッケージ作成者に付与される特権にも依存しています。

パッケージ、またはキャッシュに入った動的 SQL ステートメントがオブジェクトに依存しており、そのオブジェクトがドロップされた場合は、そのパッケージまたはキャッシュに入った動的 SQL ステートメントは、「無効」状態になります。パッケージがユーザー定義関数に従属し、その関数がドロップされると、パッケージは「作動不能」状態になります。

キャッシュに入れられた動的 SQL ステートメント (無効状態) は再び、次に使用する際に自動的に最適化されます。ステートメントに必要なオブジェクトがドロップされている場合に動的 SQL ステートメントを実行すると、失敗してエラー・メッセージが表示されることがあります。

無効状態にあるパッケージは、次に使用する際に暗黙的に再バインドされます。そのようなパッケージは明示的に再バインドすることもできます。トリガーがドロップされたためにパッケージに無効のマークが付けられた場合、再バインド・パッケージはトリガーを呼び出さなくなります。

作動不能のパッケージは、明示的に再バインドした後でなければ使用できません。

フェデレーテッド・データベースのオブジェクトには、同様の従属関係があります。たとえば、サーバーをドロップすると、そのサーバーに関連付けられたニックネームを参照するパッケージ、またはキャッシュに入れられた動的 SQL は無効になります。

ある場合には、パッケージを再バインドできないことがあります。たとえば、表がドロップされたのに再作成されない場合は、パッケージを再バインドできません。この場合、オブジェクトを再作成するか、ドロップされたオブジェクトをアプリケーションが使用しないようにアプリケーションを変更しなければなりません。

その他のほとんどの場合、たとえば、制約の一部がドロップされた場合は、パッケージを再バインドすることが可能です。

以下のシステム・カタログ・ビューは、パッケージの状態およびパッケージの従属関係を判別するのに役立ちます。

- SYSCAT.PACKAGEAUTH
- SYSCAT.PACKAGEDEP
- SYSCAT.PACKAGES

関連概念:

- 「アプリケーション開発ガイド クライアント・アプリケーションのプログラミング」の『BIND コマンドを使用したパッケージの作成』
- 「アプリケーション開発ガイド クライアント・アプリケーションのプログラミング」の『アプリケーション、バインド・ファイル、およびパッケージの関係』
- 「アプリケーション開発ガイド クライアント・アプリケーションのプログラミング」の『パッケージの再バインド』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『DROP ステートメント』
- 「*SQL リファレンス 第 1 巻*」の『SYSCAT.PACKAGEAUTH カタログ・ビュー』
- 「*SQL リファレンス 第 1 巻*」の『SYSCAT.PACKAGEDEP カタログ・ビュー』
- 「*SQL リファレンス 第 1 巻*」の『SYSCAT.PACKAGES カタログ・ビュー』
- 「コマンド・リファレンス」の『BIND コマンド』
- 「コマンド・リファレンス」の『REBIND コマンド』

第 2 部 データベースのセキュリティー

第 7 章 データベース・アクセスの制御

データベース管理者の重要な責務の 1 つに、データベースのセキュリティーがあります。データベースのセキュリティーには、いくつかの活動が含まれます。

- 装置またはシステムの誤動作によるデータまたはデータ保全性の損失を防ぐこと。
- 貴重なデータへの無許可アクセスを防ぐこと。重要な情報については、『知る必要』のない人からはアクセスできないようにしなければなりません。
- 無許可ユーザーが、誤ってデータの削除や変更を行わないように保護すること。
- 281 ページの『第 8 章 DB2 Universal Database™ (DB2 UDB) アクティビティーの監査』で説明されているユーザーによるデータのアクセスをモニターすること。

この部分では、次の点について説明します。

- 『DB2 Universal Database インストール時のセキュリティー問題』
- 230 ページの『サーバーでの認証メソッド』
- 237 ページの『リモート・クライアントの認証に関する考慮事項』
- 237 ページの『パーティション・データベースの認証に関する考慮事項』
- 278 ページの『ファイアウォール・サポートの紹介』
- 242 ページの『特権、権限レベル、およびデータベース権限』
- 260 ページの『データベース・オブジェクトへのアクセスの制御』
- 271 ページの『タスクおよび実行に必要な許可』
- 273 ページの『セキュリティー強化のためのシステム・カタログの使用』.

セキュリティーのための計画: データベース・アクセス・コントロールの目標を定義し、だれに、何を、どのような状況でアクセスさせるかを指定することから始めてください。このような計画では、データベース機能、他のプログラムの機能、および管理手順を使用することによって、これらの目的を実現する方法についても記述する必要があります。

DB2 Universal Database インストール時のセキュリティー問題

セキュリティーの問題は、製品がインストールされたときから、DB2® 管理者にとって重要なことです。

DB2 Universal Database™ (DB2 UDB) のインストールを完了するためには、ユーザー ID、グループ名、およびパスワードが必要です。 GUI ベースの DB2 UDB インストール・プログラムは、さまざまなユーザー ID とグループのデフォルト値を作成します。 UNIX® プラットフォームにインストールする場合と Windows® プラットフォームにインストールする場合とでは、作成されるデフォルト値が異なります。

- UNIX プラットフォームの場合、DB2 UDB インストール・プログラムは、異なるデフォルト・ユーザーを DAS (dasusr)、インスタンス所有者 (db2inst)、および fenced ユーザー (db2fenc) のために作成します。

  DB2 UDB インストール・プログラムは、まだ存在していないユーザー ID を作成するため、デフォルト・ユーザー名に 1 から 99 までの数値を順番に付加します。たとえば、ユーザー db2inst1 および db2inst2 がすでに存在する場合、DB2 UDB インストール・プログラムはユーザー db2inst3 を作成します。 10 よりも大きな数値が使用される場合、デフォルト・ユーザー ID の名前の文字部分が切り捨てられます。たとえば、ユーザー ID db2fenc9 がすでに存在する場合、 DB2 UDB インストール・プログラムはユーザー ID の c を切り捨てて、10 を付加します (db2fen10)。デフォルトの DAS ユーザーに数値が付加されるときに切り捨ては生じません (dasusr24 など)。
- Windows プラットフォームの場合、DB2 UDB インストール・プログラムは、デフォルト・ユーザー db2admin を DAS ユーザー、インスタンス所有者、および fenced ユーザーのために作成します。 UNIX プラットフォームの場合とは異なり、ユーザー ID に数値は付加されません。

管理者以外のユーザーがデフォルトを知って、データベースおよびインスタンス内で不適切な方式で使用するリスクを最小限にするには、インストールの際にデフォルトを新規または既存の任意のユーザー ID に変更します。

注: 応答ファイルのインストールでは、ユーザー ID またはグループ名にデフォルト値を使用しません。これらの値を、応答ファイルに指定しておく必要があります。

ユーザーを認証する際に、パスワードは非常に重要です。認証要件がオペレーティング・システム・レベルで設定されていないときに、データベースがオペレーティング・システムを使用してユーザーを認証する場合、ユーザーは接続を許可されます。たとえば、UNIX オペレーティング・システムでは、未定義のパスワードは NULL として扱われます。この場合、定義されたパスワードを持つユーザーは、NULL パスワードを持っているものと見なされます。オペレーティング・システムの観点では、これが一致であり、ユーザーの妥当性検査が行われ、データベースに接続することができます。オペレーティング・システムがデータベースに対するユーザーの認証を行う場合は、オペレーティング・システム・レベルのパスワードを使用します。

注: データベース環境を共通基準の要件に適合させる場合は、未定義のパスワードを使用できません。

DB2 Universal Database をインストールした後に、ユーザーに付与されたデフォルト特権の検討および (必要であれば) 変更も行ってください。デフォルトでは、インストール処理によってシステム管理 (SYSADM) の特権が、各オペレーティング・システム上で下記のユーザーに与えられます。

Windows 9x あらゆる Windows 98 または Windows ME ユーザー。

その他の Windows 環境 Windows NT®、Windows 2000、Windows XP、および Windows Server 2003 では、管理者グループに属する有効な DB2 UDB ユーザー名。

UNIX プラットフォーム　　インスタンス所有者の 1 次グループに属する有効な DB2 UDB のユーザー名。

SYSADM 特権は、DB2 Universal Database の中の使用可能な特権の最も強力なセットです。その結果、これらのユーザーのすべてがデフォルトによって、 SYSADM 特権を持つことを望むことはできません。 DB2 UDB は、グループおよび個々のユーザー ID に特権を与えたり、取り消したりする能力を管理者に付与しています。

データベース・マネージャーの構成パラメーター *sysadm_group* を更新することにより、管理者はどのユーザーのグループが SYSADM 特権を持つようにするかを制御できます。 DB2 UDB インストールとその後のインスタンスだけでなく、さらにデータベース作成のセキュリティー要件を完成するために、下記のガイドラインに従わなければなりません。

システム管理グループと定義される (*sysadm_group* を更新することにより) グループが、少なくとも 1 つは存在しなければなりません。このグループの名前を使用すれば、インスタンス所有者のために作成されたグループのように簡易識別することができます。このグループに属するユーザー ID およびグループは、それぞれのインスタンスに対してシステム管理者権限を持っています。

管理者は、特定インスタンスに関連付けられていると容易に認識される、インスタンス所有者ユーザー ID を作成することを考慮する必要があります。このユーザー ID は、そのグループの名前の 1 つとして上記で作成された SYSADM グループの名前を持つ必要があります。もう一つの方法としては、インスタンス所有者のユーザー ID をインスタンス所有者グループの一員としてだけに使用し、他のグループでは使用しないようにすることです。このようにすることによって、インスタンスまたはインスタンス内の任意のオブジェクトを変更できるユーザー ID およびグループの急増を制御できます。

作成されたユーザー ID は、インスタンス内のデータおよびデータベースへの入力を許可される前に認証を行うため、 1 つのパスワードと関連付ける必要があります。パスワード作成時には、編成のパスワード命名ガイドラインに従うことをお勧めします。

関連概念:

- 329 ページの『NLS 環境での命名規則』
- 330 ページの『Unicode 環境での命名規則』
- 228 ページの『Windows NT プラットフォームでのユーザーのセキュリティーに関する考慮事項』
- 229 ページの『UNIX プラットフォームでのユーザーのセキュリティーに関する考慮事項』
- 「*管理ガイド: プランニング*」の『認証』
- 「*管理ガイド: プランニング*」の『許可』
- 230 ページの『インスタンス・ディレクトリーの場所』
- 323 ページの『一般的な命名規則』
- 326 ページの『ユーザー、ユーザー ID、およびグループの命名規則』

アクセス・トークンを使用して Windows ユーザーのグループ情報を取得する

アクセス・トークンは、プロセスまたはスレッドのセキュリティー・コンテキストを説明するオブジェクトです。アクセス・トークン内の情報には、プロセスまたはスレッドに関連したユーザー・アカウントの識別および特権が含まれます。

ログオンすると、システムはユーザーのパスワードをセキュリティー・データベースに保管されている情報と比較して、それを検証します。パスワードが認証されると、システムはアクセス・トークンを生成します。ユーザーのために実行されるすべてのプロセスは、このアクセス・トークンのコピーを使用します。

アクセス・トークンは、キャッシュされた証明書に基づいて取得することもできます。システムに認証されると、その証明書はオペレーティング・システムによってキャッシュに入れられます。ドメイン・コントローラーにアクセスできないときは、キャッシュ内にある前回のログオン時のアクセス・トークンを参照できます。

アクセス・トークンには、ローカル・グループおよびさまざまなドメイン・グループ (グローバル・グループ、ドメイン・ローカル・グループ、およびユニバーサル・グループ) など、ユーザーが所属するすべてのグループに関する情報が含まれています。

注: アクセス・トークン・サポートは使用可能ですが、リモート接続を使用する場合、クライアント認証を使用するグループ・ルックアップはサポートされていません。

アクセス・トークン・サポートを使用可能にするには、 **db2set** コマンドを使用して DB2®_GRP_LOOKUP レジストリー変数を更新する必要があります。このレジストリー変数を更新する際の選択項目には、以下のものがあります。

- TOKEN

  この選択項目は、ユーザー・アカウントが定義されたロケーションで、アクセス・トークン・サポートがユーザーの所属するすべてのグループを検索できるようにします。ロケーションは通常、ドメインまたは DB2 Universal Database™ (DB2 UDB) サーバーに対してローカルな場所にあります。

- TOKENLOCAL

  この選択項目は、DB2 UDB サーバー上で、アクセス・トークン・サポートがユーザーの所属するすべてのローカル・グループを検索できるようにします。

- TOKENDOMAIN

  この選択項目は、ドメイン上で、アクセス・トークン・サポートがユーザーの所属するすべてのドメイン・グループを検索できるようにします。

アクセス・トークン・サポートを使用可能にするとき、アカウント管理インフラストラクチャーに影響を与えるいくつかの制限があります。このサポートが使用可能になると、DB2 UDB はデータベースに接続しているユーザーに関するグループ情報を収集します。 CONNECT または ATTACH 要求が成功した後でも、他の許可 ID に対する従属関係を持つ操作を行うときには、従来型のグループ列挙を使用する

必要があります。ネストされたグローバル・グループ、ドメイン・ローカル・グループ、およびキャッシュされた証明書などのアクセス・トークンの利点は、使用できません。たとえば、接続の後に他の許可 ID の下で実行するために SET SESSION_USER が使用される場合、セッションの新しい許可 ID にどの権限が付与されるかを調べるために、従来型のグループ列挙だけが使用されます。その後も、許可 ID が所属するグループに対して特権を付与することおよび取り消すこととは対照的に、 DB2 UDB に既知の個別の許可 ID に対して明示特権を付与することおよび取り消すことが必要となります。

グループを SYSADM、SYSMAINT、または SYSCTRL に割り当てようとする場合、割り当てるグループがネストされたグローバル・グループまたはドメイン・ローカル・グループではないことを確認する必要があります。その場合、キャッシュされた証明書の機能は必要ありません。

DB2_GRP_LOOKUP レジストリー変数を使用して、 DB2 UDB が従来型のグループ列挙方法を使用してどこでグループをルックアップするかを示すために、グループ・ルックアップ・ロケーションを指定することを検討してください。たとえば、次のようにします。

```
db2set DB2_GRP_LOOKUP=LOCAL,TOKENLOCAL
```

これにより、ローカル・グループを列挙するためのアクセス・トークン・サポートが使用可能になります。接続されたユーザーとは異なる許可 ID のグループ・ルックアップは、 DB2 UDB サーバーで実行されます。

```
db2set DB2_GRP_LOOKUP=,TOKEN
```

これにより、ユーザー ID が定義されたロケーションで、グループを列挙するためのアクセス・トークン・サポートが使用可能になります。接続されたユーザーとは異なる許可 ID のグループ・ルックアップは、ユーザー ID が定義された場所で実行されます。

```
db2set DB2_GRP_LOOKUP=DOMAIN,TOKENDOMAIN
```

これにより、ドメイン・グループを列挙するためのアクセス・トークン・サポートが使用可能になります。接続されたユーザーとは異なる許可 ID のグループ・ルックアップは、ユーザー ID が定義された場所で実行されます。

DYNAMICRULES RUN (デフォルト) を使用するパッケージ結合内の、動的 SQL を使用するアプリケーションは、そのアプリケーションを実行するユーザーの特権の下で実行します。この場合、前述の制限は適用されません。これには、JDBC および DB2 CLI を使用するために作成されたアプリケーションが含まれます。

アクセス・トークン・サポートは、CLIENT 認証を除くすべての認証タイプによって使用可能になります。

注: Windows® NT 4.0 ユーザーの場合、 DB2 UDB アプリケーションがローカル暗黙接続を使用しているときは、アクセス・トークン・サポートは処理レベルのセキュリティー・コンテキストだけをサポートします。つまり、アプリケーション内のすべてのスレッドが、アプリケーションを実行しているユーザーのセキュリティー・コンテキストの下で実行しているかのように処理されます。異なるスレッドに対して異なるユーザー・セキュリティー・コンテキストを必

要とする場合、 Windows 2000 以降に移動することを検討するか、または明示接続を使用するように DB2 UDB アプリケーションを変更することを検討してください。

関連概念:

- 223 ページの『DB2 Universal Database インストール時のセキュリティー問題』

オペレーティング・システムでのセキュリティーに関する詳細

オペレーティング・システムはそれぞれ、セキュリティーを管理する方法を提供しています。オペレーティング・システムに関連したセキュリティーの問題のいくつかについて、このセクションで説明します。

注: DB2 Universal Database™ (DB2 UDB) が、 SERVER_ENCRYPT 認証の使用時にユーザー ID とパスワードを暗号化するために使用する暗号ルーチンと、DATA_ENCRYPT 認証の使用時にユーザー ID、パスワード、およびユーザー・データを暗号化するために使用する暗号ルーチンは、FIPS 140-2 に準拠しています。

暗号ルーチンは、IBM Crypto for C (ICC) バージョン 1.2.1 によって提供されます。 ICC の FIPS 140-2 Validation Certificate No. 384 は、 NIST の Web サイト内の次の場所にあります。
http://csrc.nist.gov/cryptval/140-1/140crt/140crt384.pdf

セキュリティー・ポリシーも NIST の Web サイト内の次の場所にあります。
http://csrc.nist.gov/cryptval/140-1/140sp/140sp384.pdf

セキュリティー・ポリシーには、DB2 UDB を FIPS 140-2 に準拠した方法でインストールするための指針があります。詳細については、セクション 5.3.2 を参照してください。

FIPS 140-2 への準拠が可能なシステムは、 AIX、Microsoft Windows、Solaris オペレーティング環境、Linux、および HP-UX だけです。サポートされるシステムの完全な詳細については、 Validation Certificate とセキュリティー・ポリシーを参照してください。

Windows NT プラットフォームでのユーザーのセキュリティーに関する考慮事項

システム管理 (SYSADM) 権限は、そのユーザー・アカウントが定義されているマシンのローカル管理者グループに属している、有効なあらゆる DB2® Universal Database (DB2 UDB) ユーザー・アカウントに付与されます。

Windows® ドメイン環境のデフォルトでは、インスタンスに対する SYSADM 権限を付与されるのは、ドメイン・コントローラーの管理者グループに属しているドメイン・ユーザーだけです。 DB2 UDB は必ずアカウントが定義されているマシンで許可を行うので、サーバーのローカル管理者グループにドメイン・ユーザーを追加しても、そのグループにはドメイン・ユーザー SYSADM 権限は付与されません。

注: Windows NT® にあるようなドメイン環境では、DB2 UDB は、要件と制約事項を満たし、ユーザー ID が属する最初の 64 グループのみを認証します。グループは 64 より多いという可能性もあります。

PDC の管理者グループにドメイン・ユーザーが追加されないようにするには、グローバル・グループを作成し、 SYSADM 権限を付与するユーザー (ドメインとローカルの両方) を追加します。これを行うには、以下のコマンドを入力します。

```
DB2STOP
DB2 UPDATE DBM CFG USING SYSADM_GROUP global_group
DB2START
```

関連概念:

- 229 ページの『UNIX プラットフォームでのユーザーのセキュリティーに関する考慮事項』

Windows ローカル・システム・アカウントのサポート

Windows® プラットフォームで、DB2® Universal Database (DB2 UDB) は、ローカル暗黙接続があるローカル・システム・アカウント (LSA) のコンテキストで実行するアプリケーションをサポートします。このアカウントの下で実行するアプリケーションを作成する開発者は、「SYS」で始まるスキーマ名のオブジェクトに関して DB2 UDB に制約があることに注意する必要があります。そのため、アプリケーションに DB2 UDB オブジェクトを作成する DDL が含まれる場合、以下の方法で作成する必要があります。

- 静的 SQL では、QUALIFIER オプションの値をデフォルト以外のものにしてバインドする必要があります。
- 動的 SQL では、作成するオブジェクトを DB2 UDB によってサポートされるスキーマ名で明示的に修飾するか、 CURRENT SCHEMA レジスターを DB2 UDB によってサポートされるスキーマ名に設定する必要があります。

LSA のグループ情報は、DB2 UDB インスタンスが開始した後の最初のグループ・ルックアップで収集され、インスタンスが再起動するまで更新されません。

関連概念:

- 223 ページの『DB2 Universal Database インストール時のセキュリティー問題』

UNIX プラットフォームでのユーザーのセキュリティーに関する考慮事項

DB2® Universal Database (DB2 UDB) は、データベース管理者としてのルート動作を直接サポートしてはいません。データベース管理者としては **su - <instance owner>** を使用してください。

セキュリティーの理由で、インスタンス名を fenced ID として使用しないことをお勧めします。ただし、fenced UDF またはストアード・プロシージャーを使用する計画がないならば、別のユーザー ID を作成する代わりに fenced ID をインスタンス名に設定することができます。

推奨は、このグループに関連付けられていると認識されるユーザー ID を作成することです。 fenced UDF およびストアード・プロシージャーのユーザーは、インスタンス作成スクリプト (**db2icrt ... -u <FencedID>**) のパラメーターとして指定されます。 DB2 クライアントまたは DB2 Software Developer's Kit をインストールする場合、これは必須ではありません。

関連概念:

- 228 ページの『Windows NT プラットフォームでのユーザーのセキュリティーに関する考慮事項』

インスタンス・ディレクトリーの場所

UNIX® では、 **db2icrt** コマンドはインスタンス所有者のホーム・ディレクトリーの下に、メイン SQL ライブラリー (`sqllib`) ディレクトリーを作成します。

Windows® オペレーティング・システムでは、インスタンス・ディレクトリーは DB2® がインストールされたディレクトリーの、 `/sqllib` サブディレクトリーにあります。

関連概念:

- 17 ページの『インスタンスの作成』

関連タスク:

- 23 ページの『追加のインスタンスの作成』

セキュリティー・プラグイン

DB2® Universal Database (DB2 UDB) での認証は、セキュリティー・プラグイン を介して行われます。詳しくは、 「*アプリケーション開発ガイド クライアント・アプリケーションのプログラミング*」の『セキュリティー・プラグイン』を参照してください。

サーバーでの認証メソッド

インスタンスまたはデータベースにアクセスするためには、まず、そのユーザーが *認証* されていることが必要です。各インスタンスの *認証タイプ* によって、ユーザーを検査する方法と場所が決まります。認証タイプは、サーバーのデータベース・マネージャー構成ファイルに保管されます。認証タイプは、インスタンスの作成時に初期設定されます。インスタンスごとに 1 つの認証タイプがあり、それが、そのデータベース・サーバーおよびその制御下のすべてのデータベースのアクセスをカバーしています。

フェデレーテッド・データベースからデータ・ソースにアクセスしたい場合、データ・ソース認証処理およびフェデレーテッド認証タイプの定義を考慮する必要があります。

以下の認証タイプがあります。

SERVER
ローカルのオペレーティング・システムのセキュリティーを使用して、サー

バー上で認証が行われることを指定します。接続が試みられているときにユーザー ID およびパスワードが指定されると、それらがサーバーにある有効なユーザー ID とパスワードの組み合わせと比較され、そのユーザーがそのインスタンスへのアクセスを許されているかどうかが判別されます。これがデフォルトのセキュリティー・メカニズムです。

注:

1. サーバー・コードは、接続がローカルなのかリモートなのかを検出します。ローカル接続の場合、認証が SERVER であると、ユーザー ID とパスワードは、認証の成功のためには必要とされません。
2. DB2® Universal Database (DB2 UDB) をインストールして共通基準認定構成をセットアップする場合、 SERVER を指定する必要があります。

SERVER_ENCRYPT

サーバーが、暗号化された SERVER 認証スキーマを受け入れるように指定します。クライアント認証が指定されない場合、クライアントはサーバーで選択された方式を使用して認証されます。

クライアント認証が SERVER である場合、クライアントはユーザー ID およびパスワードをサーバーに渡すことによって認証されます。クライアント認証が SERVER_ENCRYPT である場合、クライアントはユーザー ID および暗号化されたパスワードを渡すことによって認証されます。

SERVER_ENCRYPT がクライアントで指定され、SERVER がサーバーで指定されると、認証レベルの不一致のためにエラーが戻されます。

CLIENT

オペレーティング・システムのセキュリティーを使用して、アプリケーションが呼び出されたデータベース・パーティション上で認証が行われることを指定します。接続が試みられているときにユーザー ID およびパスワードが指定されると、それらがクライアント・ノードにある有効なユーザー ID とパスワードの組み合わせと比較され、そのユーザー ID がそのインスタンスへのアクセスを許されているかどうかが判別されます。データベース・サーバーでは、それ以上の認証は行われません。これはしばしば、シングル・サインオンと呼ばれます。

ユーザーがローカルまたはクライアントのログインを行った場合、そのユーザーは、そのローカルのクライアント・ワークステーションでのみ認識されます。

リモート・インスタンスが CLIENT 認証である場合、 *trust_allclnts* と *trust_clntauth* という他の 2 つのパラメーターが最終的な認証タイプを決定します。

TRUSTED クライアントのみに対する CLIENT レベルのセキュリティー

トラステッド・クライアントとは、信頼できるローカル・セキュリティー・システムをもつクライアントのことです。具体的には、Windows® 9x の各オペレーティング・システムを除く、すべてのクライアントがトラステッド・クライアントです。

CLIENT の認証タイプが選択されている場合、固有のセキュリティーをオペレーティング環境が持っていないクライアントに対する保護のために、追加のオプションを選択することができます。

非セキュアのクライアントに対する保護のために、管理者は、*trust_allclnts* パラメーターを NO に設定することによって、「トラステッド・クライアント認証」を選択することができます。これは、すべてのトラステッド・プラットフォームが、サーバーに代わってユーザーの認証ができることを意味します。非トラステッド・クライアントは、サーバー上で認証され、ユーザー ID とパスワードを提供しなければなりません。ユーザーは、クライアントを信頼するかどうかを示すために、 *trust_allclnts* 構成パラメーターを使用します。このパラメーターのデフォルトは YES です。

注: 一部のクライアントが認証のためのネイティブの安全なセキュリティー・システムを持っていない場合であっても、すべてのクライアントをトラステッド・クライアント (*trust_allclnts* が YES) とすることは可能です。

トラステッド・クライアントの場合であっても、サーバー側で認証を完了させたい場合があります。トラステッド・クライアントをどこで妥当性検査するかを指示するために、 *trust_clntauth* 構成パラメーターを使用します。このパラメーターのデフォルトは CLIENT です。

注: トラステッド・クライアントの場合のみ、 CONNECT または ATTACH を試みているときにユーザー ID またはパスワードが明示的に提供されないと、ユーザーの妥当性検査は、そのクライアントで行われます。 *trust_clntauth* パラメーターは、 USER または USING 文節で提供された情報をどこで妥当性検査するかを判別するためだけに使用されます。

DRDA® クライアントを除くすべてのクライアントからの保護を、 DB2 for OS/390® and z/OS™、 DB2 for VM and VSE、および DB2 for iSeries™ から行うには、 *trust_allclnts* パラメーターを DRDAONLY に設定します。上記のクライアントだけを、クライアント側の認証を行うよう承認することができます。他のすべてのクライアントには、サーバーによって認証されているユーザー ID とパスワードが必要です。

trust_clntauth パラメーターは、上記のクライアントが認証される位置を判別するのに使用されます。 *trust_clntauth* が ″client″ である場合、認証はクライアントで行われます。 *trust_clntauth* を ″server″ に設定すると、認証は、クライアント (パスワードが指定されなかった場合) およびサーバー (パスワードが指定された場合) で行われます。

表 5. TRUST_ALLCLNTS および TRUST_CLNTAUTH パラメーターの組み合わせを使用した認証モード

TRUST_ ALLCLNTS	TRUST_ CLNTAUTH	非トラステッドである DRDA クライアント認証、パスワードなし	非トラステッドである DRDA クライアント認証、パスワードあり	トラステッドである DRDA クライアント認証、パスワードなし	トラステッドである DRDA クライアント認証、パスワードあり	DRDA クライアント認証、パスワードなし	DRDA クライアント認証、パスワードあり
YES	CLIENT	CLIENT	CLIENT	CLIENT	CLIENT	CLIENT	CLIENT
YES	SERVER	CLIENT	SERVER	CLIENT	SERVER	CLIENT	SERVER
NO	CLIENT	SERVER	SERVER	CLIENT	CLIENT	CLIENT	CLIENT
NO	SERVER	SERVER	SERVER	CLIENT	SERVER	CLIENT	SERVER
DRDAONLY	CLIENT	SERVER	SERVER	SERVER	SERVER	CLIENT	CLIENT

表 5. *TRUST_ALLCLNTS および TRUST_CLNTAUTH パラメーターの組み合わせを使用した認証モード (続き)*

TRUST_ ALLCLNTS	TRUST_ CLNTAUTH	非トラステッドである DRDA クライアント認証、パスワードなし	非トラステッドである DRDA クライアント認証、パスワードあり	トラステッドである DRDA クライアント認証、パスワードなし	トラステッドである DRDA クライアント認証、パスワードあり	DRDA クライアント認証、パスワードなし	DRDA クライアント認証、パスワードあり
DRDAONLY	SERVER	SERVER	SERVER	SERVER	SERVER	CLIENT	SERVER

KERBEROS

DB2 UDB クライアントとサーバーが両方とも、 Kerberos セキュリティー・プロトコルをサポートしているオペレーティング・システム上で実行されている場合に使用します。 Kerberos セキュリティー・プロトコルは、従来の暗号を使用して共有秘密鍵を作成することにより、サード・パーティーの認証サービスとして認証を実行します。この鍵がユーザーの証明書になり、ローカルまたはネットワーク・サービスが要求されるたびに、ユーザーの身元の確認に使用されます。この鍵を使用することにより、ネットワークを介して生のテキストでユーザー名およびパスワードを渡す必要がなくなります。 Kerberos セキュリティー・プロトコルにより、リモート DB2 UDB サーバーへの単一のサインオンを行えるようになります。 KERBEROS 認証タイプは、Windows 2000、AIX®、および Solaris オペレーティング環境を実行しているクライアントおよびサーバーでサポートされています。

Kerberos 認証は、以下のように行われます。

1. ドメイン・アカウントを使用してクライアント・マシンにログオンしているユーザーは、ドメイン・コントローラーの Kerberos 鍵配布センター (KDC) へ認証を行います。鍵配布センターはチケット許可チケット (TGT) をクライアントに発行します。
2. 接続の最初の段階で、サーバーはクライアントにターゲット・プリンシパル名を送信します。これは、DB2 UDB サーバー・サービスのサービス・アカウント名です。サーバーのターゲット・プリンシパル名、およびチケット許可チケットを使用して、クライアントはチケット許可サービス (TGS) (これもドメイン・コントローラーにあります) へサービス・チケットを要求します。クライアントのチケット許可チケット、およびサーバーのターゲット・プリンシパル名の両方が有効であれば、TGS はクライアントへサービス・チケットを発行します。データベース・ディレクトリーに記録されるプリンシパル名は、name/instance@REALM の形式で指定できるようになります。 (これは Windows 2000 と DB2 UDB バージョン 7.1 以降の組み合わせで指定可能な、現行の DOMAIN¥userID および userID@xxx.xxx.xxx.com フォーマットに追加されるものです。)
3. クライアントはこのサービス・チケットを通信チャネル (たとえば、TCP/IP) を経由してサーバーへ送信します。
4. サーバーはクライアントのサーバー・チケットの妥当性検査を行います。クライアントのサービス・チケットが有効であれば、これで認証は完了します。

クライアント・マシンでデータベースをカタログし、サーバーのターゲット・プリンシパル名と共に Kerberos 認証タイプを明示的に指定することも可能です。そうすれば、接続の最初の段階はバイパスすることができます。

ユーザー ID およびパスワードが指定されている場合、クライアントはそのユーザー・アカウントに対するチケット許可チケットを要求し、それを認証に使用します。

KRB_SERVER_ENCRYPT
サーバーが、KERBEROS 認証または暗号化された SERVER 認証スキーマを受け入れるように指定します。クライアント認証が KERBEROS である場合、クライアントは Kerberos セキュリティー・システムを使用して認証されます。クライアント認証が SERVER_ENCRYPT である場合、クライアントはユーザー ID および暗号化パスワードを使用して認証されます。クライアント認証が指定されない場合、クライアントは Kerberos が使用可能であればそれを使用し、それが使用可能でなければパスワード暗号化を使用して認証されます。その他のクライアント認証タイプでは、認証エラーが戻されます。クライアントの認証タイプを KRB_SERVER_ENCRYPT として指定することはできません。

注: Kerberos 認証タイプがサポートされているのは、Windows 2000、Windows XP、Windows Server 2003、および AIX の各オペレーティング・システム、そして Solaris オペレーティング環境を実行しているクライアントおよびサーバーだけです。また、クライアントおよびサーバー・マシンは両方とも同じ Windows ドメインに属しているか、またはトラステッド・ドメインに属していなければなりません。この認証タイプは、サーバーが Kerberos をサポートしており、クライアント・マシンのいくつか (すべてである必要はありません) が Kerberos 認証をサポートしている場合に使用してください。

DATA_ENCRYPT
サーバーは、暗号化された SERVER 認証スキーマおよびユーザー・データの暗号化を受け入れます。認証が機能する方法は、SERVER_ENCRYPT に関して示した方法と同じです。詳しくは、その認証タイプを参照してください。

この認証タイプを使用するとき、以下のユーザー・データが暗号化されます。

- SQL ステートメント
- SQL プログラム変数データ
- SQL ステートメントのサーバー処理の出力データで、データについての説明を含むもの
- 照会から生じる応答セット・データの一部またはすべて
- ラージ・オブジェクト (LOB) データ・ストリーム
- SQLDA 記述子

DATA_ENCRYPT_CMP
サーバーは、暗号化された SERVER 認証スキーマおよびユーザー・データの暗号化を受け入れます。さらに、この認証は DATA_ENCRYPT 認証タイプをサポートしていない下位製品との互換性を可能にします。これらの製品

は、SERVER_ENCRYPT 認証タイプを使って、暗号化ユーザー・データがない状態での接続を許可されます。新しい認証タイプをサポートしている製品は、これを使用する必要があります。この認証タイプは、サーバーのデータベース・マネージャー構成ファイル内だけで有効であり、 CATALOG DATABASE コマンド上で使用するときには無効です。

GSSPLUGIN
サーバーが認証を行うために GSS-API プラグインを使用するように指定します。クライアント認証が指定されていない場合、サーバーは *srvcon_gssplugin_list* データベース・マネージャー構成パラメーターにリストされている Kerberos プラグインを含む、サーバーによってサポートされているプラグインのリストをクライアントに戻します。クライアントは、クライアント・プラグイン・ディレクトリーにある最初のプラグインをリストから選択します。クライアントがリスト内のどのプラグインもサポートしない場合、そのクライアントは Kerberos 認証方式 (それが戻される場合) を使用して認証されます。クライアント認証が GSSPLUGIN 認証方式の場合、クライアントはリスト内にあるサポートされる最初のプラグインを使用して認証されます。

GSS_SERVER_ENCRYPT
サーバーが、プラグイン認証または暗号化されたサーバー認証スキーマを受け入れるように指定します。クライアント認証がプラグインを介して行われる場合、クライアントはサーバーがサポートするプラグインのリストにある、クライアントがサポートする最初のプラグインを使用して認証されます。

クライアント認証が指定されないで暗黙的接続が行われる場合 (つまり、接続が確立されるときにクライアントがユーザー ID とパスワードを供給しない場合)、サーバーはサーバーがサポートするプラグインのリスト、Kerberos 認証方式 (リスト内のプラグインの 1 つが Kerberos に基づくものである場合)、および暗号化サーバー認証方式を戻します。クライアントは、クライアント・プラグイン・ディレクトリーにある、最初のサポートされているプラグインを使用して認証されます。クライアントがリスト内のどのプラグインもサポートしない場合、そのクライアントは Kerberos 認証方式を使用して認証されます。クライアントが Kerberos 認証をサポートしない場合、そのクライアントは暗号化サーバー認証方式を使用して認証され、パスワードがないために接続が失敗します。クライアントは、 DB2 UDB が提供する Kerberos プラグインがオペレーティング・システムに対して存在するか、または Kerberos ベースのプラグインが *srvcon_gssplugin_list* データベース・マネージャー構成パラメーターに指定されている場合に、Kerberos 認証方式をサポートします。

クライアント認証を指定しないで明示接続が実行されている場合 (つまり、ユーザー ID とパスワードの両方が供給されている場合)、認証タイプは SERVER_ENCRYPT と等しくなります。

注:

1. 構成ファイル自体へのアクセスは構成ファイル内の情報によって保護されているため、認証情報を変更しているときに、誤って自分自身を自分のインスタンスか

らロックアウトしてしまわないようにしてください。以下のデータベース・マネージャー構成ファイル・パラメーターは、インスタンスへのアクセスを制御します。

- AUTHENTICATION *
- SYSADM_GROUP *
- TRUST_ALLCLNTS
- TRUST_CLNTAUTH
- SYSCTRL_GROUP
- SYSMAINT_GROUP

* は、2 つの最も重要なパラメーターを示し、これらが最も問題を引き起こす可能性があります。

このようなことが起こらないようにするために、行えることがいくつかあります。誤って自分自身を DB2 UDB システムからロックアウトしてしまった場合、すべてのプラットフォームで使用可能なフェイルセーフのオプションがあります。これは、高い特権をもったローカルのオペレーティング・システムのセキュリティー・ユーザーを使用して、通常の DB2 UDB セキュリティー検査をオーバーライドしてデータベース・マネージャー構成ファイルを更新することです。このユーザーは、常にデータベース・マネージャー構成ファイルを更新するための特権を持っており、それによって問題を訂正します。ただし、このセキュリティー上のう回は、データベース・マネージャー構成ファイルのローカルでの更新にのみ制限されています。フェイルセーフのためのユーザーは、リモートで、または他の DB2 UDB コマンドに対して使用することはできません。この特別のユーザーは、以下のように識別されます。

- UNIX® プラットフォームの場合: インスタンス所有者
- NT プラットフォームの場合: ローカル「管理者」グループに属している人
- その他のプラットフォームの場合: その他のプラットフォーム上ではローカル・セキュリティーがないため、すべてのユーザーがローカル・セキュリティー検査に合格します。

関連概念:

- 237 ページの『リモート・クライアントの認証に関する考慮事項』
- 237 ページの『パーティション・データベースの認証に関する考慮事項』
- 399 ページの『DB2 for Windows NT および Windows NT セキュリティーの紹介』

関連資料:

- 「*管理ガイド: パフォーマンス*」の『authentication - 「認証タイプ」構成パラメーター』
- 「*管理ガイド: パフォーマンス*」の『trust_allclnts - 「全クライアントのトラステッド化」構成パラメーター』
- 「*管理ガイド: パフォーマンス*」の『trust_clntauth - 「トラステッド・クライアント認証」構成パラメーター』

リモート・クライアントの認証に関する考慮事項

リモート・アクセスのためにデータベースをカタログする場合、認証タイプをデータベース・ディレクトリー項目の中に指定することができます。

DB2® Connect を使用してアクセスされるデータベースの場合、値が指定されないと、 SERVER 認証が使用されます。

認証タイプは必須ではありません。指定されない場合、クライアントはデフォルトの SERVER_ENCRYPT になります。ただし、サーバーが SERVER_ENCRYPT をサポートしていない場合は、クライアントはサーバーのサポートしている値を使用して再試行を行います。サーバーが複数の認証タイプをサポートしている場合は、クライアントはそれらの中から選択せずにエラーを戻します。エラーを戻すのは、正しい認証タイプが確実に使用されるようにするためです。この場合、クライアントはサポートされている認証タイプを使用してデータベースをカタログしなければなりません。認証タイプが指定されると、指定された値がサーバー側の値と一致した場合に、認証は即時に開始できます。不一致が検出された場合、DB2 Universal Database™ (DB2 UDB) はリカバリーを試行します。リカバリーにより、相違を調整するためにさらに多くのフローが実行されるか、または DB2 UDB がリカバリーできなければエラーになります。不一致がある場合は、サーバーにある値の方が正しいと見なされます。

関連概念:

- 230 ページの『サーバーでの認証メソッド』

パーティション・データベースの認証に関する考慮事項

パーティション・データベースでは、データベースの各区画に、同じ組のユーザーとグループが定義されていなければなりません。定義が同じでないと、ユーザーは、異なる区画で異なることを実行できるように許可されてしまうことがあります。すべての区画にわたって一貫していることが推奨されます。

関連概念:

- 230 ページの『サーバーでの認証メソッド』

Kerberos 認証についての詳細

DB2® Universal Database (DB2 UDB) は、 AIX® バージョン 5.2、Solaris オペレーティング環境バージョン 8、 Red Hat Enterprise Linux Server Advanced Server 2.1 (32 ビット Intel)、および Windows® 2000 以降のオペレーティング・システムで Kerberos 認証プロトコルをサポートします。

Kerberos サポートは、サーバーおよびクライアント認証プラグインとして使用される「IBMkrb5」という名前の GSS-API セキュリティー・プラグインとして提供されます。このライブラリーが置かれている場所は、 UNIX® および Linux では `sqllib/security{32|64}/plugin/IBM/{client|server}` ディレクトリー、 Windows では `sqllib/security/plugin/IBM{client|server}` ディレクトリーです。

注: 64 ビット Windows の場合、プラグイン・ライブラリーの名前は IBMkrb564.dll です。さらに、UNIX および Linux プラグインの実際のプラグイン・ソース・コード IBMkrb5.C が sqllib/samples/security/plugins ディレクトリーにあります。

Kerberos 認証を DB2 UDB でご使用になる前に、 Kerberos の使用方法と構成方法を十分に理解しておくことをお勧めします。

Kerberos の概要

Kerberos はサード・パーティーのネットワーク認証プロトコルで、無保護のネットワーク環境の中でユーザーを安全に認証するために、共有秘密鍵システムを使用します。 1980 年代後半に MIT (マサチューセッツ工科大学) で初めて開発された Kerberos の最新版は Kerberos 5 であり、 Internet Engineering Task Force (IETF) RFC 1510 で記述されています。 (IETF の URL は http://www.ieft.org、 RFC 1510 のタイトルは「The Kerberos Network Authentication Service (V5)」です。) 3 層システムが使用され、 Kerberos 鍵配布センター (KDC) という別個のサーバーが提供する暗号化されたチケットが、アプリケーション・サーバーとクライアントの間で交換されます (テキスト形式のユーザー ID とパスワードは交換されません)。このような暗号化されたサービス・チケット (*証明書* という) は存続期間が有限で、クライアントとサーバーしかこれを理解できません。これによって、たとえチケットがネットワークから傍受された場合でも、セキュリティー上のリスクが軽減されます。各ユーザー (Kerberos ではプリンシパル という) は、 KDC によって共有される暗号化された秘密鍵を所有します。 1 つの KDC に登録されたプリンシパルとコンピューターは、集合的にレルム と呼ばれます。

Kerberos の主な特徴であるシングル・サインオン環境では、各ユーザーが Kerberos レルム内のリソースに対して自分の身元を 1 度だけ検証します。これは、DB2 UDB において、ユーザーがユーザー ID およびパスワードを提供しなくても DB2 UDB サーバーに接続できるということです。もう 1 つの利点として、プリンシパル用の中心的なリポジトリーを 1 つだけ使用するため、ユーザー ID 管理が単純化されます。さらに、Kerberos は相互認証をサポートするため、クライアントはサーバーの身元を検証することができます。

Kerberos のセットアップ

DB2 UDB で Kerberos をサポートするには、 DB2 UDB を設定する前に、参加するすべてのマシンで Kerberos 層をインストールして正しく構成しなければなりません。これには、以下の要件が含まれます (この他にも要件が存在する可能性があります)。

1. クライアント・マシン、サーバー・マシン、およびプリンシパルがすべて同じレルムに所属するか、複数のトラステッド・レルム (Windows 用語ではトラステッド・ドメインという) に属する必要がある
2. 適切なプリンシパルを作成する
3. 該当する場合、サーバー keytab ファイルを作成する
4. 参加するすべてのマシンのシステム・クロックを同期する必要があります。 (通常、Kerberos では 5 分間のスキューが許容されます。そうしないと、証明書を取得するときに事前認証エラーが発生する可能性があります。)

Kerberos のインストールと構成についての詳細は、インストールする Kerberos 製品の資料を参照してください。

DB2 UDB で必要な設定は、接続するアプリケーションが提供する証明書に基づいて、 Kerberos セキュリティー・コンテキストを正常に作成することだけです (つまり認証)。その他の Kerberos 機能 (署名、メッセージ暗号化など) は、使用されません。さらに、可能な場合には、相互認証がサポートされます。

Kerberos の前提条件は以下のとおりです。

- AIX バージョン 5.2 および IBM® Network Authentication Service (NAS) Toolkit 1.3
- Solaris オペレーティング環境バージョン 8 および SEAM (Sun Enterprise Authentication Mechanism) と IBM NAS Toolkit 1.3
- Red Hat Enterprise Linux Advanced Server 2.1 および `krb5-libs` ファイルセットと `krb5-workstation` ファイルセット
- Windows 2000 (以降のオペレーティング・システムの) Server

Kerberos とクライアント・プリンシパル

プリンシパルは、2 つまたは 3 つの部分から成るフォーマットに従います (つまり、 *name@REALM* または *name/instance@REALM*)。「name」の部分は許可 ID (AUTHID) マッピングに使用されるため、この名前は DB2 UDB 命名規則に従う必要があります。つまり、名前の長さは最大 30 文字で、使用可能な文字に関する現在の制限に従う必要があります。 (AUTHID マッピングについては、後で説明します。)

注: Windows では、 Kerberos プリンシパルがドメイン・ユーザーに直接関連付けられます。このため、ドメインまたはレルムに関連付けられていない Windows マシンでは、 Kerberos 認証を使用できません。さらに、 Windows では 2 部からなる名前 (つまり *name@domain*) だけがサポートされます。

プリンシパル自体は、ターゲット・データベースへのサービス・チケットの要求および受信に使用されるアウトバウンド証明書を取得できなければなりません。取得するには、通常、UNIX または Linux では **kinit** コマンドを使用します。 Windows ではログオン時に暗黙的に取得されます。

Kerberos と許可 ID マッピング

オペレーティング・システムのユーザー ID の有効範囲が 1 つのマシンに限定される (ただし NIS は例外) のとは異なり、 Kerberos プリンシパルは自身のレルム以外のレルムでも認証可能です。プリンシパルにレルム名を付けて完全修飾することにより、プリンシパル名の重複の問題を避けることができます。 Kerberos では、完全修飾されたプリンシパルの形式は `name/instance@REALM` です。区切り文字 「/」を使ってインスタンス・フィールドに複数のインスタンスを含めることができます (たとえば、 `name/instance1/instance2@REALM`)。または、インスタンス・フィールドをまったく省略することもできます。明確な制限として、レルム名は、ネットワーク内に定義されたすべてのレルムの中でユニークでなければなりません。 DB2 UDB にとって問題となるのは、プリンシパルから AUTHID への単純なマッピングを実現するために、プリンシパル名 (完全修飾されたプリンシパルの中の「name」) と AUTHID の間で 1 対 1 のマッピングが望ましいことです。 DB2 UDB では

AUTHID がデフォルト・スキーマとして使用され、簡単かつ論理的に派生する必要があるので、単純なマッピングが必要とされます。このため、データベース管理者は、以下の問題が発生する可能性があることに注意しなければなりません。

- 異なるレルムに属する同じ名前の複数のプリンシパルは、同じ AUTHID にマップされます。
- 同じ名前でインスタンスが異なる複数のプリンシパルは、同じ AUTHID にマップされます。

この点を考慮して、以下のようにすることをお勧めします。

- DB2 UDB サーバーにアクセスするすべてのトラステッド・レルム内の名前用として、 1 つのユニークなネーム・スペースを維持する
- 同じ名前のすべてのプリンシパルを、インスタンスにかかわらず、同じユーザーに所属させる

Kerberos とサーバー・プリンシパル

UNIX および Linux では、 DB2 UDB インスタンスのサーバー・プリンシパル名は `<インスタンス名>/<完全修飾ホスト名>@REALM` と想定されます。このプリンシパルは Kerberos セキュリティー・コンテキストを受け入れる必要があり、 DB2 UDB インスタンスの開始前にすでに存在している必要があります。これは、初期化の際にサーバー名がプラグインによって DB2 UDB に報告されるためです。

Windows では、サーバー・プリンシパルは、 DB2 UDB サービスが開始されたドメイン・アカウントであると想定されます。ただし、ローカル SYSTEM アカウントによってインスタンスが開始される場合は例外です。この場合、サーバー・プリンシパル名は `host/<hostname>` と報告されます。これは、クライアントとサーバーの両方が Windows ドメインに属している場合にのみ有効です。

Windows では、2 つより多い部分からなる名前はサポートされません。このため、Windows クライアントが UNIX サーバーに接続しようとするときに問題が発生する可能性があります。したがって、UNIX Kerberos との相互運用が必要な場合、Kerberos プリンシパルから Windows アカウントへのマッピングを Windows ドメイン内で設定する必要があるかもしれません。 (関連情報については、該当する Microsoft® 資料を参照してください。)

Kerberos keytab ファイル

セキュリティー・コンテキスト要求を受け入れる UNIX または Linux 上のすべての Kerberos サービスは、証明書を *keytab* (鍵テーブル) ファイル内に格納する必要があります。これは、DB2 UDB によってサーバー・プリンシパルとして使用されるすべてのプリンシパルに当てはまります。デフォルト keytab ファイルでのみ、サーバーの鍵が検索されます。 keytab ファイルに鍵を追加する方法については、Kerberos 製品に付属の資料を参照してください。

Windows には keytab ファイルの概念がなく、システムがプリンシパルの証明書の保管および獲得を自動的に処理します。

Kerberos とグループ

Kerberos 認証プロトコルには、グループ化の概念がありません。このため、DB2 UDB は Kerberos プリンシパルのグループ・リストを取得するためにローカル・オペレーティング・システムに依存します。 UNIX または Linux の場合、各プリンシパルごとに同等のシステム・アカウントが存在しなければなりません。たとえば、プリンシパルが `name@REALM` である場合、DB2 UDB は、オペレーティング・システム・ユーザー *name* が属するすべてのグループ名をローカル・オペレーティング・システムに対して照会することにより、グループ情報を集めます。オペレーティング・システム・ユーザーが存在しない場合、 AUTHID は PUBLIC グループにのみ所属します。一方、Windows では、ドメイン・アカウントが Kerberos プリンシパルに自動的に関連付けられるため、別のオペレーティング・システム・アカウントを作成するための追加のステップは必要ありません。

クライアントでの Kerberos 認証の使用可能化

データベース・マネージャー構成パラメーター *clnt_krb_plugin* を、使用する Kerberos プラグインの名前に更新する必要があります。サポートされるプラットフォームでは、これを `IBMkrb5` に設定してください。このパラメーターは、DB2 UDB に対して、 AUTHENTICATION パラメーターが KERBEROS または KRB_SERVER_ENCRYPT に設定されていれば、 DB2 UDB で接続およびローカル・インスタンス・レベルの操作に Kerberos を使用できることを通知します。そうでない場合、クライアント・サイドの Kerberos サポートは想定されません。

注: Kerberos サポートが利用可能かどうかの検証は実行されません。

オプションで、クライアント上でデータベースのカタログを作成する場合には、認証タイプを以下のように指定できます。

```
db2 catalog db testdb at node testnode authentication kerberos target
   principal service/host@REALM
```

ただし、認証情報が提供されない場合、サーバーはサーバー・プリンシパルの名前をクライアントに送ります。

サーバーでの Kerberos 認証の使用可能化

データベース・マネージャー構成パラメーター *srvcon_gssplugin_list* を、サーバー Kerberos プラグイン名に更新する必要があります。複数のサポートされるプラグインからなるリストをこのパラメーターに含めることもできますが、 Kerberos プラグインは 1 つだけ指定してください。ただし、このフィールドがブランクで、AUTHENTICATION が KERBEROS または KRB_SERVER_ENCRYPT に設定されている場合には、デフォルト Kerberos プラグイン (`IBMkrb5`) が想定され、使用されます。 Kerberos 認証をすべての操作で使用するか、それとも着信接続だけで使用するかに応じて、パラメーター AUTHENTICATION または SVRCON_AUTH を KERBEROS または KRB_SERVER_ENCRYPT に設定してください。

Kerberos プラグインの作成

Kerberos プラグインを作成するとき、以下の点を考慮する必要があります。

- GSS-API プラグインとして Kerberos プラグインを作成します。例外として、初期化関数で DB2 UDB に戻される関数ポインター配列内の *plugintype* は、DB2SEC_PLUGIN_TYPE_KERBEROS に設定されなければなりません。
- 特定の条件のもとでは、サーバーがサーバー・プリンシパル名をクライアントに報告する場合があります。このため、プリンシパル名を GSS_C_NT_HOSTBASED_SERVICE 形式 (`service@host`) で指定しないでください。 DRDA® では、プリンシパル名が GSS_C_NT_USER_NAME 形式 (`server/host@REALM`) でなければならないためです。
- 通常は、KRB5_KTNAME 環境変数によってデフォルト keytab ファイルを指定することが可能です。ただし、サーバー・プラグインは DB2 UDB エンジン・プロセス内で実行されるため、この環境変数にアクセスできない可能性があります。

関連概念:

- 230 ページの『サーバーでの認証メソッド』

特権、権限レベル、およびデータベース権限

特権 は、ユーザーがデータベース・リソースを作成したりデータベース・リソースにアクセスしたりすることを許可するためのものです。*権限レベル* によって、特権のグループ分けの方法、およびより高いレベルのデータベース・マネージャーの保守とユーティリティー操作が得られます。*データベース権限* は、ユーザーがデータベース・レベルのアクティビティーを実行できるようにします。特権、権限レベル、およびデータベース権限を組み合わせて使用することで、データベース・マネージャーとそのデータベース・オブジェクトへのアクセスを制御できます。ユーザーは、必要な特権、権限レベル、またはデータベース権限が与えられているオブジェクトに対してのみアクセスできます。 DB2® Universal Database (DB2 UDB) は、認証されるユーザーに関してこれらの許可検査を実行します。

データベース・マネージャーでは、特定のタスクを実行するのに必要なデータベース機能を使用するために、各ユーザーが特定の許可を暗黙または明示的に与えられていなければなりません。*明示的* な権限あるいは特権は、ユーザーに対して付与されます (データベース・カタログでは GRANTEETYPE が U)。*暗黙* の権限あるいは特権は、各ユーザーが所属するグループに対して付与されます (データベース・カタログでは GRANTEETYPE が G)。たとえば、表を作成するには、表作成のための許可がユーザーに必要です。表を変更するには、表の変更を許可されていなければなりません。

243 ページの図 3 は、権限とその制御の範囲 (データベース、データベース・マネージャー) の間の関係を示します。

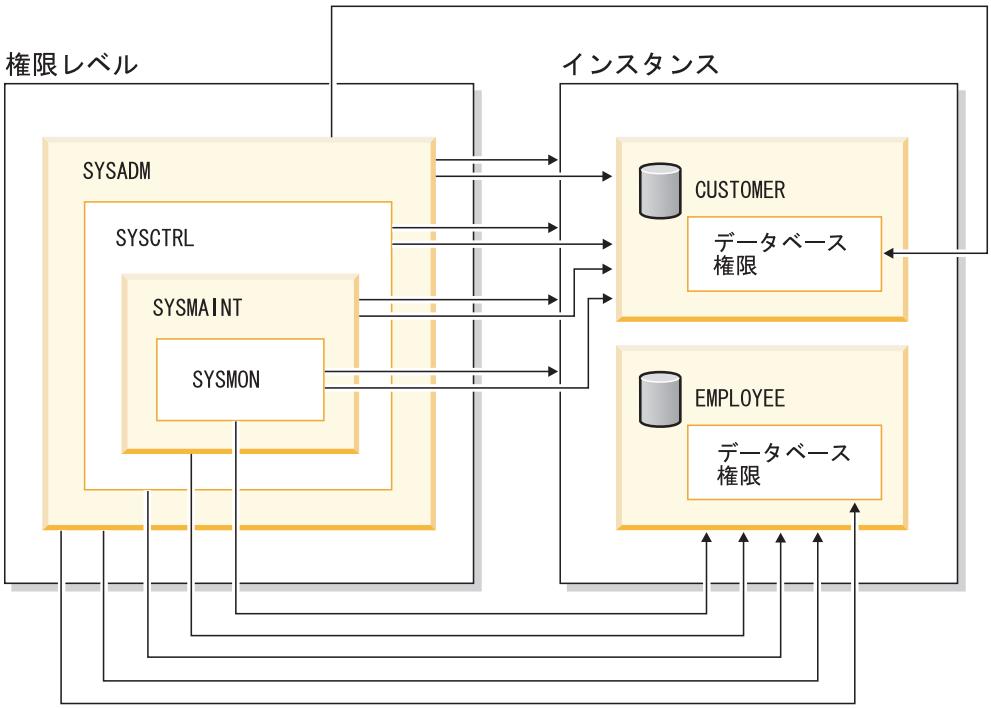


図 3. 権限の階層

各ユーザーまたはグループは、以下のような 1 つまたは複数の権限または特権を持つことができます。

- 管理権限:
  - SYSADM (システム管理者)

    SYSADM 権限レベルは、データベース・マネージャーによって作成および保守されるすべてのリソースに対する制御を与えます。システム管理者は DBADM、SYSCTRL、SYSMAINT、および SYSMON 権限をすべて所有し、DBADM 権限を付与または取り消す権限を持っています。

    SYSADM 権限を持つユーザーは、データベース・マネージャーの制御、およびデータの保護と保全性を担当します。 SYSADM 権限を持っている場合、データベース内のすべてのオブジェクトに対する暗黙特権が与えられ、どのユーザーがデータベース・マネージャーにアクセスできるか、およびそのアクセスの程度を制御することができます。 SYSADM 権限についての詳細は、 「システム管理権限 (SYSADM)」を参照してください。
  - DBADM (データベース管理者)

    DBADM データベース権限は、1 つのデータベースに対する管理権限を与えます。このデータベース管理者は、オブジェクトの作成、データベース・コマンドの発行、および表データへのアクセスに必要な権限を所有します。また、データベース管理者は、 CONTROL や個々の特権を付与または取り消すことができます。 DBADM 権限についての詳細は、 「データベース管理権限 (DBADM)」を参照してください。
- システム制御権限:
  - SYSCTRL (システム制御)

SYSCTRL 権限レベルは、システム・リソースに影響を与える操作に対する制御を可能にします。たとえば、SYSCTRL 権限を持つユーザーは、データベースの作成、更新、停止、またはドロップを行うことができます。さらに、このユーザーはインスタンスの停止を行うことができますが、表データへのアクセスはできません。 SYSCTRL 権限を持つユーザーには、SYSMON もまた与えられます。 SYSCTRL 権限についての詳細は、 「システム制御権限 (SYSCTRL)」を参照してください。

- SYSMAINT (システム保守)

  SYSMAINT 権限レベルは、インスタンスに関連したすべてのデータベースに対する保守操作を実行するのに必要な権限を与えます。 SYSMAINT 権限を持つユーザーは、データベースの更新と構成、データベースまたは表スペースのバックアップ、既存のデータベースのリストア、およびデータベースのモニターを行うことができます。 SYSCTRL と同様に、SYSMAINT は表データへのアクセス権限を与えません。 SYSMAINT 権限を持つユーザーには、SYSMON 権限もまた与えられます。 SYSMAINT 権限についての詳細は、 「システム保守権限 (SYSMAINT)」を参照してください。

- SYSMON (システム・モニター権限)

  SYSMON 権限レベルは、データベース・システム・モニターの使用に必要な権限を与えます。 SYSMON 権限についての詳細は、 「システム・モニター権限 (SYSMON)」を参照してください。

- データベース権限

  表やルーチンの作成、表へのデータのロードなどのアクティビティーを実行するには、特定のデータベース権限が必要です。詳しくは、 「データベース権限」を参照してください。

- 特権:

  データベース・オブジェクトに対するアクティビティー (たとえば索引の作成やドロップ) を実行するには、特権が必要です。特権は、ユーザーが実行できるタスクを厳密に定義します。たとえば、あるユーザーに対して、表に索引を作成する特権を与える一方、同じ表に対するトリガーを作成する特権を与えないことが可能です。

  - CONTROL 特権

    オブジェクトに対する CONTROL 特権を持っているユーザーは、そのデータベース・オブジェクトにアクセスでき、そのオブジェクトに対する他のユーザーの特権を付与または取り消すことができます。

    注: CONTROL 特権は、表、ビュー、ニックネーム、索引、およびパッケージにのみ適用されます。

    他のユーザーがそのオブジェクトに対する CONTROL 特権を要求した場合、SYSADM または DBADM 権限を持つユーザーが、そのオブジェクトに対する CONTROL 特権を付与する必要があります。 CONTROL 特権は、オブジェクト所有者から取り消されることがありません。

場合によっては、オブジェクトの作成者がそのオブジェクトに対する CONTROL 特権を自動的に取得します。詳しくは、 「オブジェクト作成、所有権、および特権」を参照してください。

- ユーザーが特定オブジェクトに対して特定のタスクを実行できるようにするために、個別特権を与えることができます。

  管理権限 (SYSADM または DBADM) を持つユーザー、または CONTROL 特権を持つユーザーは、他のユーザーの特権を付与または取り消すことができます。

  個別特権およびデータベース権限は特定の機能の実行を許可しますが、同じ特権または権限を他のユーザーに与えることはできません。 GRANT ステートメントで WITH GRANT OPTION を使用すれば、表、ビュー、スキーマ、パッケージ、ルーチン、シーケンスに関する特権を他のユーザーに対して GRANT できる権利を、他のユーザーに拡張して与えることができます。ただし、WITH GRANT OPTION を使用する場合、特権を GRANT する人が、いったん GRANT された特権を取り消すことはできません。特権を取り消すためには、SYSADM 権限、DBADM 権限、または CONTROL 特権を持っていなければなりません。

  さらに、PUBLIC に対して特権を付与することもできます。 PUBLIC 特権は、個々のユーザーにすでに特権が与えられているかどうかにかかわらず、すべてのユーザー (許可名) に適用されます。これには、将来のすべてのユーザーも含まれます。

- 暗黙特権。これは、パッケージを実行する特権を持つユーザーに与えられるものです。ユーザーがアプリケーションを実行できる場合でも、パッケージ内で使用されるデータ・オブジェクトに対する明示特権が必要であるとは限りません。

1 つのユーザーまたはグループに対して、個々の特権または権限をいくつか組み合わせて許可することもできます。特権をオブジェクトに関連付ける場合、そのオブジェクトはすでに存在していなければなりません。たとえば、表がそれ以前に作成されているのでなければ、その表についての SELECT 特権をユーザーに与えることはできません。

注: ある許可名が権限と特権を与えられ、しかもその許可名で作成されたユーザーがいない場合には、注意が必要です。後で、その許可名を使用してユーザーが作成され、その許可名に関連するすべての権限と特権を自動的に受け取る可能性があります。

すでに付与された特権を取り消すには、REVOKE ステートメントを使用します。DB2 UDB では、1 つの許可名から特権を取り消すと、すべての許可名によって付与された特権が取り消されます。

ある許可名から特権を取り消しても、その許可名によって特権を付与された他の許可名からその同じ特権が取り消されることはありません。たとえば、ユーザー CLAIRE が SELECT WITH GRANT OPTION をユーザー RICK に与えた後、RICK が SELECT を BOBBY および CHRIS に与えたとします。もし CLAIRE が SELECT 特権を RICK から取り消しても、 BOBBY と CHRIS は引き続き SELECT 特権を保持します。

関連概念:

- 248 ページの『システム管理権限 (SYSADM)』
- 248 ページの『システム制御権限 (SYSCTRL)』
- 249 ページの『システム保守権限 (SYSMAINT)』
- 250 ページの『データベース管理権限 (DBADM)』
- 252 ページの『LOAD 権限』
- 252 ページの『データベース権限』
- 255 ページの『スキーマの特権』
- 256 ページの『表スペース特権』
- 256 ページの『表およびビューの特権』
- 258 ページの『パッケージの特権』
- 259 ページの『索引の特権』
- 259 ページの『シーケンス特権』
- 260 ページの『データベース・オブジェクトへのアクセスの制御』
- 265 ページの『パッケージ経由の間接特権』
- 260 ページの『ルーチン特権』
- 246 ページの『オブジェクト作成、所有権、および特権』
- 251 ページの『システム・モニター権限 (SYSMON)』

オブジェクト作成、所有権、および特権

オブジェクトが作成されるとき、1 つの許可名に対して、そのオブジェクトの*所有権* が割り当てられます。所有権を与えられているユーザーは、任意の SQL ステートメントを使ってそのオブジェクトを参照することを許可されます。

スキーマ内でオブジェクトを作成するとき、ステートメントの許可 ID は、暗黙的または明示的に指定されるスキーマ内でオブジェクトを作成するのに必要な特権を持っていなければなりません。つまり、許可名がスキーマの所有者であるか、スキーマに対する CREATEIN 特権を持っている必要があります。

注: 表スペース、バッファー・プール、またはデータベース・パーティション・グループを作成するときには、この要件は適用されません。これらのオブジェクトはスキーマ内には作成されません。

オブジェクトが作成されるとき、ステートメントの許可 ID がそのオブジェクトの所有者になります。

注: ただし、1 つの例外があります。 CREATE SCHEMA ステートメントで AUTHORIZATION オプションを指定した場合、 CREATE SCHEMA 操作の一部として作成されるすべてのオブジェクトは、 AUTHORIZATION オプションが指定する許可 ID によって所有されます。ただし、最初の CREATE SCHEMA 操作の後でスキーマ内で作成されるすべてのオブジェクトは、特定の CREATE ステートメントに関連した許可 ID によって所有されます。

たとえば、ステートメント CREATE SCHEMA SCOTTSTUFF AUTHORIZATION SCOTT CREATE TABLE T1 (C! INT) によって、スキーマ SCOTTSTUFF および表 SCOTTSTUFF.T1 が作成され、このどちらもユーザー SCOTT によって所有されます。ここで、ユーザー BOBBY に対して SCOTTSTUFF スキーマに対する CREATEIN 特権が与えられ、 BOBBY が表 SCOTTSTUFF.T1 への索引を作成するとします。索引はスキーマの後で作成されるため、 SCOTTSTUFF.T1 への索引を所有するのは BOBBY です。

特権は、作成されるオブジェクトのタイプに応じて、以下のようにオブジェクト所有者に割り当てられます。

- CONTROL 特権は、新しく作成される表、索引、およびパッケージに対して暗黙的に付与されます。この特権を持つオブジェクト作成者は、そのデータベース・オブジェクトにアクセスでき、そのオブジェクトに対する他のユーザーの特権を付与または取り消すことができます。他のユーザーがそのオブジェクトに対する CONTROL 特権を要求した場合、 SYSADM または DBADM 権限を持つユーザーが、そのオブジェクトに対する CONTROL 特権を付与する必要があります。オブジェクト所有者は、CONTROL 特権を取り消すことができません。
- ビュー定義によって参照されるすべての表、ビュー、およびニックネームに対する CONTROL 特権をオブジェクト所有者が持っている場合、新しく作成されるビューに対して CONTROL 特権が暗黙的に付与されます。
- 他のオブジェクト (トリガー、ルーチン、シーケンス、表スペース、バッファー・プールなど) には、 CONTROL 特権が関連付けられません。オブジェクト所有者は、オブジェクトに関連付けられるすべての特権を自動的に受け取ります (さらに所有者は、サポートされている場合、GRANT ステートメントで WITH GRANT オプションを使用することで、これらの特権を他のユーザーに与えることができます)。また、オブジェクト所有者は、オブジェクトの変更、コメントの追加、およびオブジェクトのドロップを行うことができます。これらの許可はオブジェクト所有者に暗黙的に与えられ、取り消すことはできません。

関連概念:

- 242 ページの『特権、権限レベル、およびデータベース権限』
- 255 ページの『スキーマの特権』
- 256 ページの『表スペース特権』
- 256 ページの『表およびビューの特権』
- 258 ページの『パッケージの特権』
- 259 ページの『索引の特権』
- 259 ページの『シーケンス特権』
- 260 ページの『ルーチン特権』

特権、権限、および許可に関する詳細

このセクションでは、各権限について説明し、次にさまざまな特権について述べます。

システム管理権限 (SYSADM)

SYSADM 権限レベルは、最も高いレベルの管理権限です。 SYSADM 権限を与えられたユーザーは、ユーティリティーを実行したり、データベースおよびデータベース・マネージャーのコマンドを発行したり、データベース・マネージャー・インスタンス内のデータベースの表データにアクセスしたりできます。この権限は、インスタンス内のすべてのデータベース・オブジェクトを制御します。制御されるデータベース・オブジェクトには、データベース、表、ビュー、索引、パッケージ、スキーマ、サーバー、別名、データ・タイプ、関数、プロシージャー、トリガー、表スペース、データベース・パーティション・グループ、バッファー・プール、およびイベント・モニターがあります。

SYSADM 権限は、 *sysadm_group* 構成パラメーターによって指定されたグループに割り当てられます。このグループのメンバーシップは、データベース・マネージャーの外で、プラットフォームで使われているセキュリティー機能によって制御されます。

SYSADM 権限を持つユーザーだけが実行できる機能は、次のとおりです。

- データベースの移行
- データベース・マネージャー構成ファイルの変更 (SYSCTRL、SYSMAINT、または SYSMON 権限のあるグループを指定することを含む)
- DBADM 権限の GRANT

注: SYSADM 権限を持つユーザーがデータベースを作成した場合、そのユーザーには、データベースに対する DBADM 権限が明示的に付与されます。データベース作成者を SYSADM グループから除去する場合、このユーザーがそのデータベースに DBADM としてアクセスできないようにするには、ユーザーの DBADM 権限を明示的に取り消す必要があります。

関連概念:

- 248 ページの『システム制御権限 (SYSCTRL)』
- 249 ページの『システム保守権限 (SYSMAINT)』
- 270 ページの『データ暗号化』
- 251 ページの『システム・モニター権限 (SYSMON)』

システム制御権限 (SYSCTRL)

SYSCTRL 権限は、最も高いレベルのシステム制御権限です。この権限があると、データベース・マネージャーのインスタンスとそのデータベースに対して、保守およびユーティリティー操作を実行することができます。これらの操作はシステム・リソースに影響を及ぼす場合がありますが、データベース内のデータに対するアクセスは認められていません。システム制御権限は、重要データの入ったデータベース・マネージャーのインスタンスを管理するユーザーを対象としたものです。

SYSCTRL 権限は、 *sysctrl_group* 構成パラメーターによって指定されたグループに割り当てられます。グループが指定されると、そのグループのメンバーシップは、プラットフォーム上で使用されるセキュリティー機能によって、データベース・マネージャーの外で制御されます。

SYSCTRL 以上の権限を持つユーザーだけが実行できることは、次のとおりです。

- データベース、ノード、または分散接続サービス (DCS) ディレクトリーの更新
- システムからのユーザーの切断
- データベースの作成またはドロップ
- 表スペースのドロップ、作成、または変更
- 新しいデータベースへのリストア

さらに、SYSCTRL 権限を持つユーザーは、システム保守権限 (SYSMAINT) およびシステム・モニター権限 (SYSMON) を持つユーザーの機能を実行できます。

SYSCTRL 権限を持つユーザーは、データベースへの接続に関する暗黙の特権も持っています。

注: SYSCTRL 権限を持つユーザーがデータベースを作成すると、そのユーザーには、そのデータベースに対する明示的な DBADM 権限が自動的に付与されます。データベースの作成者が SYSCTRL グループから除去され、そのデータベースに DBADM としてアクセスすることも防止したい場合は、この DBADM 権限を明示的に取り消さなければなりません。

関連概念:

- 249 ページの『システム保守権限 (SYSMAINT)』
- 250 ページの『データベース管理権限 (DBADM)』
- 251 ページの『システム・モニター権限 (SYSMON)』

システム保守権限 (SYSMAINT)

SYSMAINT 権限は、2 番目のレベルのシステム制御権限です。この権限があると、データベース・マネージャー・インスタンスとそのデータベースに対して、保守およびユーティリティー操作を実行することができます。これらの操作はシステム・リソースに影響を及ぼす場合がありますが、データベース内のデータに対するアクセスは認められていません。システム保守権限は、重要データの入ったデータベース・マネージャー・インスタンス内のデータベースを保守するユーザーを対象としています。

SYSMAINT 権限は、 *sysmaint_group* 構成パラメーターによって指定されたグループに割り当てられます。グループが指定されると、そのグループのメンバーシップは、プラットフォーム上で使用されるセキュリティー機能によって、データベース・マネージャーの外で制御されます。

SYSMAINT 以上の権限を持つユーザーだけが実行できることは、次のとおりです。

- データベースの構成ファイルの更新
- データベースまたは表スペースのバックアップ
- 既存のデータベースへのリストア
- ロールフォワード・リカバリーの実行
- インスタンスの開始または停止
- 表スペースのリストア
- トレースの実行

- データベース・マネージャー・インスタンスまたはそのデータベースのデータベース・システム・モニター・スナップショットの取得

SYSMAINT、DBADM、またはそれ以上の権限を持つユーザーは、次のことを実行できます。

- 表スペースの状態の照会
- ログ履歴ファイルの更新
- 表スペースの静止
- 表の再編成
- **RUNSTATS** ユーティリティーを使用してのカタログ統計の収集

さらに、SYSMAINT 権限を持つユーザーは、データベースに接続する特権を暗黙的に与えられ、システム・モニター権限 (SYSMON) を持つユーザーに許可された機能を実行することもできます。

関連概念:

- 250 ページの『データベース管理権限 (DBADM)』
- 251 ページの『システム・モニター権限 (SYSMON)』

データベース管理権限 (DBADM)

DBADM 権限は、2 番目にレベルの高い管理権限です。この権限は特定のデータベースにのみ適用され、ユーザーは、特定のユーティリティーを実行し、データベース・コマンドを出し、そしてデータベース内のどの表のデータにもアクセスすることができます。 DBADM 権限が付与されると、 BINDADD、 CONNECT、 CREATETAB、 CREATE_EXTERNAL_ROUTINE、 CREATE_NOT_FENCED_ROUTINE、 IMPLICIT_SCHEMA、 QUIESCE_CONNECT、および LOAD データベース権限もまた与えられます。 SYSADM 権限を持つユーザーだけが DBADM 権限の付与または取り消しを実行できます。 DBADM 権限を持つユーザーは、データベースに対する特権を他のユーザーに付与できます。また、だれが特権を付与したかにかかわりなく、ユーザーの特権を取り消すこともできます。

DBADM 以上の権限を持つユーザーだけが実行できることは、次のとおりです。

- ログ・ファイルの読み取り
- イベント・モニターの作成、活動化、およびドロップ

DBADM、SYSMAINT、またはそれ以上の権限を持つユーザーは、次のことを実行できます。

- 表スペースの状態の照会
- ログ履歴ファイルの更新
- 表スペースの静止
- 表の再編成
- **RUNSTATS** ユーティリティーを使用してのカタログ統計の収集

注: DBADM によって上記の機能を実行できるのは、 DBADM 権限が与えられているデータベースに対してだけです。

関連概念:

- 248 ページの『システム管理権限 (SYSADM)』
- 248 ページの『システム制御権限 (SYSCTRL)』
- 249 ページの『システム保守権限 (SYSMAINT)』
- 252 ページの『LOAD 権限』
- 252 ページの『データベース権限』
- 254 ページの『暗黙スキーマ権限 (IMPLICIT_SCHEMA) に関する考慮事項』

システム・モニター権限 (SYSMON)

SYSMON 権限は、データベース・マネージャー・インスタンスまたはそのデータベースを対象とするデータベース・システム・モニター・スナップショットの取得を許可します。 SYSMON 権限は、構成パラメーター *sysmon_group* によって指定されたグループに割り当てられます。グループが指定されると、そのグループのメンバーシップは、プラットフォーム上で使用されるセキュリティー機能によって、データベース・マネージャーの外で制御されます。

SYSMON 権限を持つユーザーは、以下のコマンドを実行できます。

- GET DATABASE MANAGER MONITOR SWITCHES
- GET MONITOR SWITCHES
- GET SNAPSHOT
- LIST ACTIVE DATABASES
- LIST APPLICATIONS
- LIST DCS APPLICATIONS
- RESET MONITOR
- UPDATE MONITOR SWITCHES

SYSMON 権限を持つユーザーは、以下の API を使用できます。

- db2GetSnapshot - スナップショットの取得
- db2GetSnapshotSize - db2GetSnapshot() 出力バッファーに必要なサイズの見積もり
- db2MonitorSwitches - モニター・スイッチの入手/更新
- db2ResetMonitor - モニターのリセット

SYSMON 権限を持つユーザーは、以下の SQL 表関数を使用できます。

- すべての表スナップショット関数 (あらかじめ `SYSPROC.SNAPSHOT_FILEW` を実行する必要はありません)

  `SYSPROC.SNAPSHOT_FILEW` はスナップショットを取得して、その内容をファイルに保管します。ヌルの入力パラメーターを使って表スナップショット関数を呼び出した場合、リアルタイムのシステム・スナップショットの代わりに、ファイルの内容が戻されます。

SYSADM、SYSCTRL、または SYSMAINT 権限レベルを持つユーザーには、SYSMON 権限もまた与えられます。

関連資料:

- 「*管理ガイド: パフォーマンス*」の『sysmon_group - システム・モニター権限グループ名構成パラメーター』

LOAD 権限

データベース・レベルの LOAD 権限、および表に対する INSERT 特権を持っているユーザーは、 **LOAD** コマンドを使用してデータを表にロードすることができます。

データベース・レベルの LOAD 権限、および表に対する INSERT 特権を持っているユーザーは、直前のロード操作でデータを挿入するロードを行った場合に、**LOAD RESTART** または **LOAD TERMINATE** を行うことができます。

データベース・レベルの LOAD 権限、および表に対する INSERT 特権と DELETE 特権を持っているユーザーは、 **LOAD REPLACE** コマンドを使用できます。

直前のロード操作でロード置換を行った場合、ユーザーは、DELETE 特権が付与されていないと、 **LOAD RESTART** または **LOAD TERMINATE** を行うことができません。

ロード操作の一部として例外表が使用される場合、ユーザーには、その例外表に対する INSERT 特権が必要です。

この権限を持っているユーザーは、 **QUIESCE TABLESPACES FOR TABLE**、**RUNSTATS**、および **LIST TABLESPACES** コマンドを実行することができます。

関連概念:

- 「*データ移動ユーティリティー ガイドおよびリファレンス*」の『ロードの使用に必要な特権、権限、および許可』
- 256 ページの『表およびビューの特権』

関連資料:

- 「コマンド・リファレンス」の『RUNSTATS コマンド』
- 「コマンド・リファレンス」の『QUIESCE TABLESPACES FOR TABLE コマンド』
- 「コマンド・リファレンス」の『LIST TABLESPACES コマンド』
- 「コマンド・リファレンス」の『LOAD コマンド』

データベース権限

253 ページの図 4 は、データベース権限を示しています。

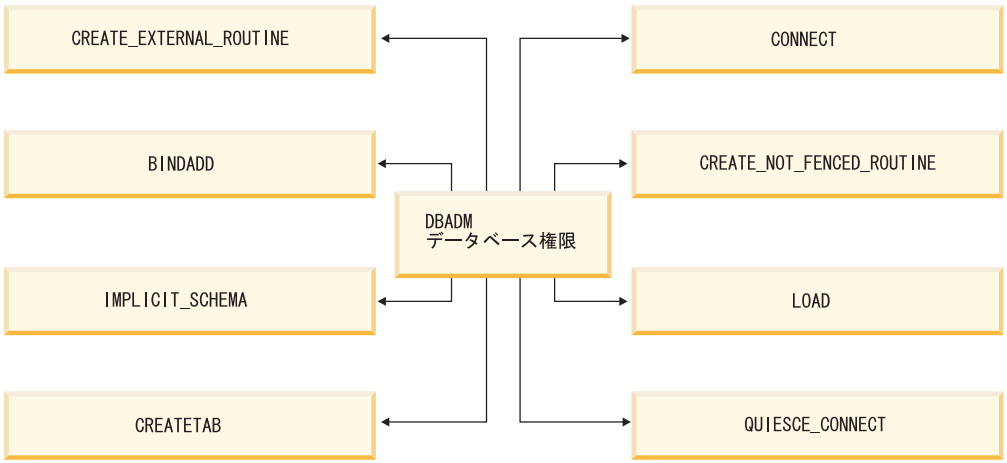


図 4. データベース権限

データベース権限は、データベース全体に対するアクションを許可します。 DBADM 権限を持つすべてのユーザーは、以下のようなデータベース権限のセットをすべて所有しています。

- CONNECT: ユーザーはデータベースに接続できます。
- BINDADD: ユーザーはデータベースに新しいパッケージを作成できます。
- CREATETAB: ユーザーはデータベースに新しい表を作成できます。
- CREATE_EXTERNAL_ROUTINE: ユーザーは、アプリケーションによって、またデータベースの他のユーザーによって使用されるプロシージャーを作成できます。
- CREATE_NOT_FENCED_ROUTINE: ユーザーは、「fenced でない」のユーザー定義関数 (UDF) またはプロシージャーを作成できます。「fenced でない」の UDF またはプロシージャーは、特に十分にテストしなければなりません。これは、データベース・マネージャーはそのストレージや制御ブロックを、これらの UDF またはプロシージャーから保護しないためです。 (このため、「fenced でない」での実行が許される UDF またはプロシージャーの作成およびテストが不十分であると、システムに重大な問題が起きることがあります。)

  注: CREATE_EXTERNAL_ROUTINE は、 CREATE_NOT_FENCED_ROUTINE を持つすべてのユーザーに対して自動的に付与されます。
- IMPLICIT_SCHEMA: どのユーザーも、すでに存在していないスキーマ名を指定した CREATE ステートメントを使用してオブジェクトを作成することによって、暗黙にスキーマを作成することができます。 SYSIBM が暗黙に作成されたスキーマの所有者になり、 PUBLIC にこのスキーマ内にオブジェクトを作成するための特権が与えられます。
- LOAD ユーザーは表にデータをロードできます。
- QUIESCE_CONNECT ユーザーは静止中のデータベースに接続できます。

他のユーザーにデータベース権限を付与したり他のユーザーからデータベース権限を取り消すことができるのは、 SYSADM または DBADM 権限を持つユーザーだけです。

注: データベース作成時に、以下のデータベース権限が自動的に PUBLIC に付与されます。

- CREATETAB データベース権限
- BINDADD データベース権限
- CONNECT データベース権限
- IMPLICIT_SCHEMA データベース権限
- USERSPACE1 表スペースに対する USE 特権
- システム・カタログ・ビュー上での SELECT 特権

データベース権限を除去するためには、 DBADM または SYSADM が明示的に PUBLIC からデータベース権限を取り消さなければなりません。

関連タスク:

- 261 ページの『特権の付与』
- 262 ページの『特権の取り消し』

暗黙スキーマ権限 (IMPLICIT_SCHEMA) に関する考慮事項

新しいデータベースが作成されるとき、 PUBLIC に IMPLICIT_SCHEMA データベース権限が与えられます。この権限を使用して、どのユーザーも、オブジェクトを作成し、すでに存在していないスキーマ名を指定することによって、スキーマを作成することができます。 SYSIBM が暗黙に作成されたスキーマの所有者になり、 PUBLIC にこのスキーマ内にオブジェクトを作成するための特権が与えられます。

だれが暗黙にスキーマ・オブジェクトを作成できるかを制御することがデータベースで必要な場合は、 IMPLICIT_SCHEMA データベース権限を PUBLIC から取り消す必要があります。いったんこれを行うと、スキーマ・オブジェクトが作成される方法は、以下の 3 つしかありません。

- どのユーザーも、CREATE SCHEMA ステートメントで自分自身の許可名を使用してスキーマを作成することができます。
- DBADM 権限を持つどのユーザーも、すでに存在していなければどのスキーマでも明示的に作成することができ、オプションで、別のユーザーをそのスキーマの所有者として指定することができます。
- DBADM 権限をもつどのユーザーも (PUBLIC と独立して) IMPLICIT_SCHEMA データベース権限を持っているため、他のデータベース・オブジェクトを作成しているときに、任意の名前を持ったスキーマを暗黙に作成することができます。 SYSIBM が暗黙に作成されたスキーマの所有者になり、 PUBLIC がスキーマ内にオブジェクトを作成する特権を持ちます。

関連タスク:

- 261 ページの『特権の付与』
- 262 ページの『特権の取り消し』

スキーマの特権

スキーマ特権は、オブジェクト特権区分に入ります。オブジェクト特権は、図 5 に示されています。

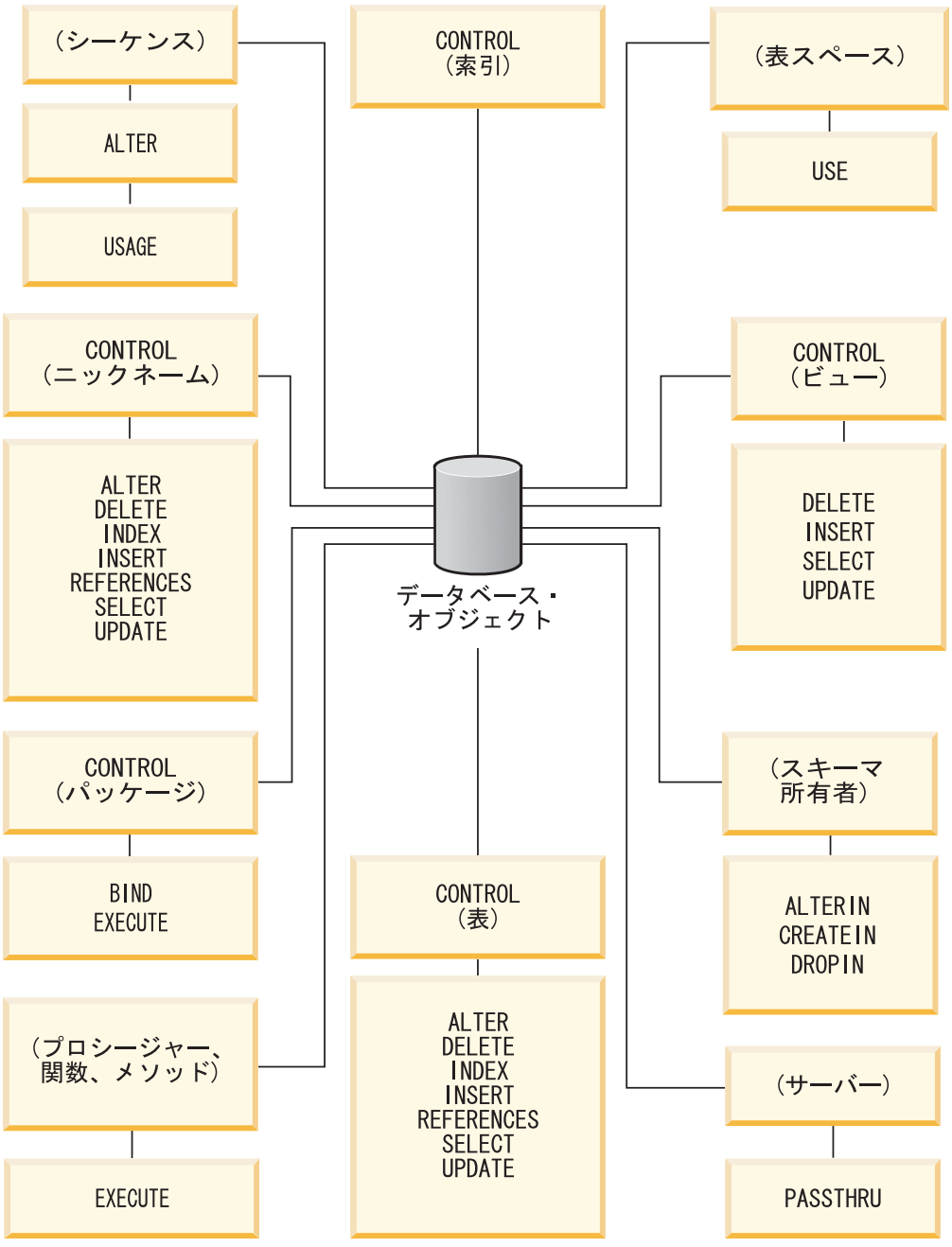


図 5. オブジェクト特権

スキーマの特権には、データベース内のスキーマ上でのアクションが含まれます。ユーザーには、以下の特権のどれでも付与することができます。

- CREATEIN により、ユーザーはスキーマ内にオブジェクトを作成できます。
- ALTERIN により、ユーザーはスキーマ内のオブジェクトを変更できます。
- DROPIN により、ユーザーはスキーマ内からオブジェクトをドロップできます。

スキーマの所有者は、これらの特権をすべて持ち、その特権を他のユーザーに付与する能力を持ちます。スキーマ・オブジェクト内で操作されるオブジェクトには、表、ビュー、索引、パッケージ、データ・タイプ、関数、トリガー、プロシージャー、および別名があります。

関連タスク:

- 261 ページの『特権の付与』
- 262 ページの『特権の取り消し』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『ALTER SEQUENCE ステートメント』

表スペース特権

表スペース特権は、データベースの表スペースに対してアクションを実行することを可能にします。表スペースの USE 特権を付与されたユーザーは、その表スペース内で表を作成できます。

表スペースの所有者 (多くの場合、 SYSADM または SYSCTRL 権限を持っている作成者) は USE 特権を持っており、他のユーザーにこの特権を付与することができます。デフォルトでは、データベースの作成時に、表スペース USERSPACE1 に関する USE 特権が PUBLIC に付与されますが、この特権は取り消すこともできます。

USE 特権は、 SYSCATSPACE または SYSTEM TEMPORARY 表スペースでは使用できません。

関連タスク:

- 261 ページの『特権の付与』
- 262 ページの『特権の取り消し』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『CREATE TABLE ステートメント』

表およびビューの特権

表およびビューの特権には、データベース内の表やビューに対するアクションが含まれます。次に挙げる特権のいずれかを使用するユーザーには、データベースについての CONNECT 権限が必要です。

- CONTROL 特権は、表やビューのドロップ、表についての個別の特権の GRANT や REEVOKE を含む、表やビューに対する全ての特権を許可します。
  CONTROL 特権を付与するには、SYSADM または DBADM 権限が必要です。表の作成者は、自動的にその表の CONTROL 特権を受け取ります。ビューの作成者は、ビュー定義の中で参照されているすべての表、ビュー、およびニックネームに対する CONTROL 特権を持っているか、 SYSADM または DBADM 権限を持っている場合にのみ、自動的に CONTROL 特権を受け取ります。
- ALTER は、ユーザーが表を変更することを許可します (たとえば、表への列またはユニーク制約の追加)。 ALTER 特権を持つユーザーは、表または表の列に対す

る COMMENT ON も可能です。表に対する実行可能な変更操作については、ALTER TABLE および COMMENT ステートメントの説明を参照してください。

- DELETE 特権は、表やビューからの行の削除をユーザーに許可します。
- INDEX 特権は、表についての索引の作成をユーザーに許可します。索引の作成者には、索引についての CONTROL 特権が自動的に与えられます。
- INSERT 特権は、表やビューに対する行の挿入や、 IMPORT ユーティリティーの実行をユーザーに許可します。
- REFERENCES 特権は、表に対するリレーションの親としての指定や、外部キーの作成および削除をユーザーに許可します。ユーザーは、特定の列にのみこの特権を持つことができます。
- SELECT 特権は、表やビューからの行の取り出しや、表に基づくビューの作成、EXPORT ユーティリティーの実行をユーザーに許可します。
- UPDATE 特権は、表またはビューの項目の変更、あるいは表またはビューの 1 つ以上の特定の列の中の項目の変更をユーザーに許可します。ユーザーは、特定の列にのみこの特権を持つことができます。

GRANT ステートメントの WITH GRANT OPTION を使用して、これらの特権を他のユーザーに GRANT する特権を GRANT することもできます。

注: ユーザーまたはグループが、ある表の CONTROL 特権を GRANT された場合、その表に対する他のすべての特権は、自動的に WITH GRANT OPTION によって GRANT されます。その後、表に対する CONTROL 特権をユーザーから取り消しても、そのユーザーは自動的に付与された他の特権を依然として持っています。 CONTROL 特権と一緒に付与された特権をすべて取り消す場合は、特権を個別に明示的に取り消すか、または REVOKE ステートメントに ALL キーワードを指定しなければなりません。以下にその例を示します。

```
REVOKE ALL
   ON EMPLOYEE FROM USER HERON
```

タイプ表に関しては、表およびビューの特権に関連したインプリメンテーションがあります。

注: 特権は、表階層の各レベルで別々に付与されます。その結果、タイプ表の階層内のスーパー表で特権を付与されたユーザーは、副表にも間接的に影響を与えることがあります。しかし、その副表で必要な特権が保持されている場合は、ユーザーは副表に対する操作を直接的にしか行えません。

表階層の表の間のスーパー表/副表関係は、SELECT、UPDATE、および DELETE などの操作が、操作のターゲット表とそのすべての副表 (あれば) の行に影響を与えることを意味します。この性質を*代替性*と呼ぶことができます。たとえば、タイプ Manager_t の副表マネージャーを使用して、タイプ Employee_t の Employee 表を作成したとします。構造型 Employee_t と Manager_t 間のタイプ/サブタイプ関係、また表 Employee と Manager 間の対応する表/副表関係によって示されているとおり、マネージャーはある種の (特殊な) 従業員です。この関係の結果、次の SQL 照会は、

```
SELECT * FROM Employee
```

従業員とマネージャー両方のオブジェクト ID と Employee_t 属性を戻します。同様に、次の更新操作は、

```
UPDATE Employee SET Salary = Salary + 1000
```

マネージャーと従業員の給与を 1000 ドル引き上げます。

Employee 表で SELECT 特権を持つユーザーは、 Manager 表で明示的な SELECT 特権を持っていなくても、この SELECT 操作を実行できます。しかし、そのようなユーザーは、 Manager 副表に対して SELECT 操作を直接実行することは許可されませんので、 Manager 表の継承されたのではない列にアクセスすることはできません。

同様に、Employee 表で UPDATE 特権を持つユーザーは、Manager 表で明示的な UPDATE 特権がなくても、通常の従業員とマネージャー両方に影響を与えるような、Manager に対する UPDATE 操作を実行できます。しかし、そのようなユーザーは、 Manager 副表に対して UPDATE 操作を直接実行することは許可されませんので、 Manager 表の継承されたのではない列を更新することはできません。

関連概念:

- 259 ページの『索引の特権』

関連タスク:

- 261 ページの『特権の付与』
- 262 ページの『特権の取り消し』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『ALTER TABLE ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『CREATE VIEW ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『SELECT ステートメント』

パッケージの特権

パッケージとは、データベース・オブジェクトの 1 つで、データベース・マネージャーが、特定のアプリケーション・プログラムにとって最も効率的な方法でデータにアクセスするのに必要な情報が入っています。パッケージの特権を与えられたユーザーは、パッケージの作成と操作を行うことができます。以下のいずれかの特権を使用するユーザーには、データベースに対する CONNECT 権限が必要です。

- CONTROL 特権は、パッケージの再バインド、ドロップ、または実行、およびこれらの特権を他のユーザーに与えることをユーザーに許可します。パッケージの作成者には自動的にこの特権が与えられます。 CONTROL 特権を持つユーザーには、BIND 特権と EXECUTE 特権が付与されます。さらに、そのユーザーは GRANT ステートメントを使って他のユーザーにこれらの特権を付与することもできます。 (WITH GRANT OPTION を使用して特権を GRANT すれば、 BIND または EXECUTE 特権を受け取るユーザーは、その特権をさらに他のユーザーに与えることができます。) CONTROL 特権を付与するためには、SYSADM または DBADM 権限が必要です。
- パッケージの BIND 特権は、そのパッケージの再バインドやバインド、また、同じパッケージ名と作成者の新規のバージョンの追加をユーザーに許可します。

- EXECUTE 特権は、パッケージの実行をユーザーに許可します。

注: すべてのパッケージ特権は、パッケージ名と作成者が同じすべてのバージョンに適用されます。

これらのパッケージ特権に加えて、BINDADD データベース特権によって、ユーザーは、データベース内に新しいパッケージを作成するか、または既存のパッケージを再バインドすることができます。

ニックネームによって参照されるオブジェクトは、オブジェクトを格納するデータ・ソースでの認証検査をパスする必要があります。さらに、パッケージ・ユーザーには、データ・ソースにあるデータ・ソース・オブジェクトに対する適切な特権、または権限レベルが必要です。

DB2 ファミリーのデータ・ソースを使って通信するとき、 DB2® Universal Database (DB2 UDB) は動的 SQL を使用するため、ニックネームを含むパッケージが付加的な許可ステップを必要とする可能性があります。データ・ソースでパッケージを実行する許可 ID には、そのデータ・ソースでパッケージを動的に実行するための適切な権限が必要です。

関連概念:

- 252 ページの『データベース権限』

関連タスク:

- 261 ページの『特権の付与』
- 262 ページの『特権の取り消し』

索引の特権

索引または索引の指定の作成者には、索引に対する CONTROL 特権が自動的に与えられます。索引の CONTROL 特権は、実際には、索引を削除することを可能にします。ある索引の CONTROL 特権を付与するためには、そのユーザーには SYSADM または DBADM 権限が必要です。

表レベルの INDEX 特権は、表に関する索引の作成をユーザーに許可します。

ニックネーム・レベルの INDEX 特権は、そのニックネームに対する索引指定の作成をユーザーに許可します。

関連概念:

- 256 ページの『表およびビューの特権』

関連タスク:

- 261 ページの『特権の付与』
- 262 ページの『特権の取り消し』

シーケンス特権

シーケンスの作成者には、そのシーケンスに対する USAGE および ALTER 特権が自動的に与えられます。 USAGE 特権は、シーケンスに対して NEXT VALUE お

よび PREVIOUS VALUE 式を使用するために必要とされます。他のユーザーに NEXT VALUE および PREVIOUS VALUE 式の使用を許可するには、シーケンスの USAGE 特権を PUBLIC に付与しなければなりません。これで、すべてのユーザーが指定されたシーケンスでこれらの式を使用できるようになります。

シーケンスに対する ALTER 特権を持つユーザーは、シーケンスの再始動、今後のシーケンス値の増分の変更といったタスクを実行できます。シーケンスの作成者は ALTER 特権を他のユーザーに GRANT でき、 WITH GRANT OPTION を使用すれば、それらのユーザーもまた、これらの特権をさらに他のユーザーに GRANT できるようになります。

関連タスク:

- 261 ページの『特権の付与』
- 262 ページの『特権の取り消し』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『ALTER SEQUENCE ステートメント』

ルーチン特権

EXECUTE 特権には、データベース内の関数、プロシージャー、およびメソッドといったすべてのタイプのルーチンのアクションが含まれます。 EXECUTE 特権を与えられたユーザーはルーチンを呼び出すことができ、そのルーチンからのソース関数を作成できます (関数にのみ適用されます)。また、CREATE VIEW、CREATE TRIGGER といった任意の DDL ステートメント内でルーチンを参照できます。

外部のストアード・プロシージャー、関数、またはメソッドを定義するユーザーは、 EXECUTE WITH GRANT 特権を付与されます。 WITH GRANT OPTION を使用して EXECUTE 特権を他のユーザーに GRANT した場合、そのユーザーは、さらに他のユーザーに EXECUTE 特権を GRANT できます。

関連タスク:

- 261 ページの『特権の付与』
- 262 ページの『特権の取り消し』

データベース・オブジェクトへのアクセスの制御

データベース・アクセスを制御するには、直接および間接の特権、管理権限、およびパッケージについての理解が必要です。この部分ではそうした点について説明し、いくつかの例を挙げます。

直接付与される特権は、システム・カタログに格納されます。

権限許可を制御する方法には、次の 3 通りのものがあります。

- 明示的な許可は、 GRANT および REVOKE ステートメントを使用して制御される特権によって制御されます。
- 暗黙の許可は、オブジェクトの作成とドロップによって制御されます。
- 間接特権はパッケージに関連したものです。

注: GRANT または REVOKE ステートメント、またはコントロール・センターで使用するときのデータベース・グループ名は、 8 文字以下でなければなりません。 8 文字より長いデータベース・グループ名も受け入れられますが、そのような長い名前の場合は、そのグループに属するユーザーがデータベース・オブジェクトにアクセスするときに、エラー・メッセージが出されます。

関連概念:

- 273 ページの『セキュリティー強化のためのシステム・カタログの使用』

関連タスク:

- 261 ページの『特権の付与』
- 262 ページの『特権の取り消し』

データベース・オブジェクトへのアクセス・コントロールに関する詳細

データベース・オブジェクトへのアクセス・コントロールは、 GRANT ステートメントおよび REVOKE ステートメントの使用によって行われます。暗黙的なアクセス許可および間接特権についても説明します。

特権の付与

制約事項:

ほとんどのデータベース・オブジェクトに対する特権を GRANT するには、ユーザーがそのオブジェクトに対する SYSADM 権限、DBADM 権限、または CONTROL 特権を持つか、またはそのユーザーが WITH GRANT OPTION 特権を保持していなければなりません。特権を付与できるのは既存のオブジェクトについてだけです。他ユーザーに CONTROL 特権を付与するためには、SYSADM または DBADM 権限が必要です。 DBADM 権限を付与するには、SYSADM 権限が必要です。

手順:

GRANT ステートメントは、許可ユーザーが特権を付与することができるようにするものです。特権は、1 つのステートメントで、1 つ以上の許可名に付与するか、あるいは、特権をすべてのユーザーが使用可能なようにする PUBLIC に付与することができます。許可名は、個別のユーザーかまたはグループのいずれかにすることができることに注意してください。

オペレーティング・システムに同じ名前のユーザーとグループがある場合、ユーザーとグループのどちらに特権を付与するのかを指定する必要があります。 GRANT および REVOKE ステートメントのどちらにおいても、 USER および GROUP というキーワードがサポートされています。これらのオプションのキーワードが使用されない場合、データベース・マネージャーはオペレーティング・システムのセキュリティー機能をチェックして、その許可名がユーザーであるかグループであるかを判別します。許可名がユーザーとグループの両方である可能性がある場合、エラーが戻されます。

次の例は、HERON というユーザーに対して、 EMPLOYEE 表についての SELECT 特権を付与するものです。

```
GRANT SELECT
   ON EMPLOYEE TO USER HERON
```

次の例は、HERON というグループに対して、 EMPLOYEE 表についての SELECT 特権を付与するものです。

```
GRANT SELECT
   ON EMPLOYEE TO GROUP HERON
```

関連概念:

- 260 ページの『データベース・オブジェクトへのアクセスの制御』

関連タスク:

- 262 ページの『特権の取り消し』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『GRANT (データベース権限) ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『GRANT (索引特権) ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『GRANT (パッケージ特権) ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『GRANT (スキーマ特権) ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『GRANT (表、ビュー、またはニックネーム特権) ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『GRANT (サーバー特権) ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『GRANT (表スペース特権) ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『GRANT (シーケンス特権) ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『GRANT (ルーチン特権) ステートメント』

特権の取り消し

REVOKE ステートメントを使用すれば、許可ユーザーは、他のユーザーに付与されている特権を取り消すことができます。

制約事項:

データベース・オブジェクトに対する特権を取り消すには、DBADM 権限、SYSADM 権限、またはそのオブジェクトに対する CONTROL 特権が必要です。WITH GRANT OPTION 特権を保持しているだけでは、その特権を取り消すには十分でないことに注意してください。他のユーザーの CONTROL 特権を取り消すには、SYSADM または DBADM 権限が必要です。 DBADM 権限を取り消すには、SYSADM 権限が必要です。特権を取り消すことができるのは、既存のオブジェクトについてだけです。

注: DBADM 権限または CONTROL 特権を持っていないユーザーは、 WITH GRANT OPTION を使用して GRANT した特権を取り消すことはできません。また、取り消された人から付与された特権を受け取っているユーザーに対する取り消しには、連鎖はありません。

明示的に付与された表 (またはビュー) に対する特権が DBADM 権限によってユーザーから取り消される場合、その表に定義されている他のビューに対する特権は取

り消されることは**ありません**。ビューに対する特権は DBADM 権限によって使用可能になるものであり、基本表の明示特権とは関係ないからです。

手順:

同じ名前のユーザーとグループの両方に特権が付与されている場合、特権を取り消す時に、GROUP と USER キーワードのどちらかを指定する必要があります。次の例は、HERON というユーザーの EMPLOYEE 表に対する SELECT 特権を取り消すものです。

```
REVOKE SELECT
   ON EMPLOYEE FROM USER HERON
```

次の例は、HERON というグループの EMPLOYEE 表に対する SELECT 特権を取り消すものです。

```
REVOKE SELECT
   ON EMPLOYEE FROM GROUP HERON
```

1 つのグループから特権を取り消しても、そのグループに属するすべてのメンバーからその特権が取り消されるとは限らないことに注意してください。個別の名前が特権を直接付与されている場合は、その特権が直接取り消されるまで保持されます。

表特権がユーザーから取り消される場合、取り消された表特権に依存するそのユーザーによって作成されたすべてのビューに対する特権も取り消されます。ただし、システムによって暗黙に付与された特権のみが取り消されます。ビューに対する特権が別のユーザーによって直接付与された場合、その特権は引き続き保持されます。

特権をグループに付与してから、そのグループの 1 人のメンバーだけから特権を取り消すという状況があります。エラー・メッセージ SQL0556N を受け取らないでこれを行うには、次の 2 つの方法だけを行ってください。

- グループからそのメンバーを除去します。あるいは、メンバーを減らして新規グループを作成し、その新規グループに特権を付与します。
- グループから特権を取り消してから、個々のユーザー (許可 ID) に特権を付与します。

注: 表またはビューに対する CONTROL 特権がユーザーから取り消された場合でも、そのユーザーは、その特権を他のユーザーに付与する能力は持ち続けます。 CONTROL 特権が与えられると、そのユーザーは他の WITH GRANT OPTION 特権もすべて受け取ります。 CONTROL が取り消されても、他の特権のすべては、それらが明示して取り消されるまで、 WITH GRANT OPTION のまま残されます。

取り消された特権に依存しているすべてのパッケージは無効と見なされますが、十分な権限を持つユーザーによって再バインドされると再び有効になります。特権が後で再びアプリケーションをバインドしたユーザーに付与される場合、パッケージも再作成することができます。そのアプリケーションを実行すると、暗黙の再バインドが正常に実行されるトリガーとなります。特権が PUBLIC から取り消された場合、 PUBLIC 特権に基づいたバインドしかできないユーザーによってバインドされていたすべてのパッケージが無効にされます。ユーザーの持つ DBADM 権限が取り

消されると、そのユーザーによってバインドされたパッケージはすべて無効になります。データベース・ユーティリティーに関連するパッケージも例外ではありません。無効のマークが付けられているパッケージを使用しようとすると、システムは、そのパッケージの再バインドを試みます。この再バインドの試みが失敗すると、エラー (SQLCODE -727) が発生します。この場合、それらのパッケージを明示的に再バインドするには、以下の権限が必要です。

- それらのパッケージを再バインドするための権限
- それらのパッケージ内で使われているオブジェクトに対する該当する権限

そうしたパッケージの再バインドは、特権を取り消す時に行うべきです。

1 つまたは複数の特権に基づいてトリガーまたは SQL 関数を定義した場合、これらの特権のいくつかを失うと、そのトリガーまたは SQL 関数を使用できなくなります。

関連タスク:

- 261 ページの『特権の付与』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『REVOKE (データベース権限) ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『REVOKE (索引特権) ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『REVOKE (パッケージ特権) ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『REVOKE (スキーマ特権) ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『REVOKE (表、ビュー、またはニックネーム特権) ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『REVOKE (サーバー特権) ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『REVOKE (表スペース特権) ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『REVOKE (ルーチン特権) ステートメント』

オブジェクトの作成とドロップによる暗黙許可の管理

手順:

データベース・マネージャーは、表、パッケージなどのデータベース・オブジェクトを作成するユーザーに対して、いくつかの特権を暗黙的に付与します。特権は、SYSADM または DBADM 権限を持つユーザーによってオブジェクトが作成されるときにも付与されます。同様に、オブジェクトをドロップすると特権はドロップされます。

作成されるオブジェクトが、表、ニックネーム、索引、またはパッケージであれば、ユーザーにはそのオブジェクトに対する CONTROL 特権が与えられます。オブジェクトがビューの場合、そのビューに対する CONTROL 特権が暗黙のうちに付与されるのは、ユーザーがそのビュー定義の中で参照されるすべての表、ビュー、およびニックネームに対する CONTROL 特権を持っている場合に限られます。

明示的に作成されたオブジェクトがスキーマである場合、そのスキーマの所有者には、 WITH GRANT OPTION によって ALTERIN、CREATEIN、および DROPIN 特権が与えられます。暗黙に作成されたスキーマは、PUBLIC に付与された CREATEIN 特権を持ちます。

関連タスク:

- 261 ページの『特権の付与』
- 262 ページの『特権の取り消し』

パッケージの所有権の確立

手順:

BIND および PRECOMPILE コマンドは、アプリケーション・パッケージを作成または変更します。どちらのコマンドでも、 OWNER オプションを使って結果パッケージの所有者の名前を付けてください。パッケージの所有者の命名には、単純なルールがあります。

- どのユーザーも、自分を所有者として命名できます。 OWNER オプションが指定されていない場合、これがデフォルトです。
- SYSADM または DBADM 権限を持つ ID は、OWNER オプションを使って、任意の許可 ID を所有者として命名することができます。

DB2™ (DB2 UDB) データベース製品を使用するパッケージをバインドできるすべてのオペレーティング・システムが、 OWNER オプションをサポートしているわけではありません。

関連資料:

- 「コマンド・リファレンス」の『BIND コマンド』
- 「コマンド・リファレンス」の『PRECOMPILE コマンド』

パッケージ経由の間接特権

アプリケーション・プログラムによって、および対話式ワークステーション・セッションを操作しているユーザーによって、データベース内のデータに対するアクセスが要求されることがあります。パッケージに含まれているステートメントによって、ユーザーは多数のデータベース・オブジェクトに対して様々なアクションを実行できます。そのような各アクションには、1 つまたは複数の特権が必要です。

パッケージをバインドしている個別ユーザーに付与された特権、および PUBLIC に付与された特権は、静的 SQL がバインドされるときの許可検査のために使用されます。グループを通して付与された特権は、静的 SQL がバインドされるときの許可検査には使用されません。有効な *authID* を持ち、パッケージをバインドするユーザーは、パッケージのバインド時に VALIDATE RUN が指定されている場合を除いて、そのパッケージ内の静的 SQL ステートメントを実行するために必要なすべての特権を明示的に付与されているか、あるいは PUBLIC を通して必要な特権が暗黙に付与されているかのいずれかでなければなりません。 BIND 時に VALIDATE RUN を指定すると、そのパッケージ内のどの静的 SQL ステートメントに許可の障害が発生したとしても BIND は失敗せず、その SQL ステートメントはランタイム

に再び有効になります。ユーザーがパッケージをバインドするための、適切な許可 (BIND または BINDADD 特権) を持っているかどうか検査を行う場合には、PUBLIC、グループ、およびユーザーの特権がすべて使用されます。

パッケージには、静的 SQL および動的 SQL の両方を含めることができます。静的 SQL を含むパッケージを処理する場合、ユーザーに必要なのはパッケージについての EXECUTE 特権だけです。したがってそのユーザーは、パッケージ内の静的 SQL に対するパッケージ・バインド・プログラムの特権を間接的に取得することができます。ただし、これはパッケージによって課される制限の範囲内に限られます。

動的 SQL がパッケージに含まれる場合、必要な特権は、パッケージのプリコンパイル時またはバインド時に DYNAMICRULES に指定された値に応じて異なります。詳しくは、動的 SQL に対する DYNAMICRULES の影響について説明したトピックを参照してください。

関連概念:

- 266 ページの『ニックネームが定義されているパッケージ経由の間接特権』
- 「*アプリケーション開発ガイド クライアント・アプリケーションのプログラミング*」の『動的 SQL における DYNAMICRULES BIND オプションの影響』

関連資料:

- 「*コマンド・リファレンス*」の『BIND コマンド』

ニックネームが定義されているパッケージ経由の間接特権

パッケージにニックネームへの参照が含まれる場合、パッケージ作成者およびパッケージ・ユーザーの許可処理はもう少し複雑です。パッケージ作成者が、ニックネームを含むパッケージを正常にバインドする場合、パッケージ作成者は、ニックネームがデータ・ソースで参照する表およびビューに関して、許可検査または特権検査をパスする必要はありません。しかし、パッケージの実行者は、データ・ソースで認証および許可検査をパスすることが必要です。

たとえば、パッケージ作成者の .SQC ファイルに、複数の SQL ステートメントが入っているとします。 1 つの静的ステートメントはローカル表を参照します。別の動的ステートメントはニックネームを参照します。パッケージがバインドされると、ローカル表およびニックネームの特権を検証するためにパッケージ作成者の許可 ID が使用されます。しかし、ニックネームが識別するデータ・ソース・オブジェクトに関しては何も検査されません。別のユーザーが、そのパッケージ用の EXECUTE 特権があることを前提としてパッケージを実行する場合、そのユーザーは表を参照するステートメントへの付加的な特権検査をパスする必要はありません。しかし、ニックネームを参照するステートメントの場合は、パッケージを実行するユーザーはデータ・ソースで認証検査と特権検査をパスすることが必要です。

.SQC ファイルに、動的 SQL ステートメントと、表とニックネームの参照の混合だけが入っている場合、ローカル・オブジェクトとニックネームへの DB2® Universal Database (DB2 UDB) 許可検査は同じです。パッケージ・ユーザーは、ステートメント内のすべてのローカル・オブジェクト (表、ビュー) に関する特権検査をパスしなければならず、さらにニックネーム・オブジェクトの特権検査もパスする必要があります (パッケージ・ユーザーは、ニックネームが識別するオブジェクトを含む

データ・ソースで認証および特権検査をパスしなければなりません)。どちらの場合も、パッケージのユーザーには EXECUTE 特権が必要です。

パッケージの ID およびパスワードは、すべてのデータ・ソース認証、および特権処理に使用されます。この情報は、ユーザー・マッピングの作成によって変更できます。

注: ニックネーム経由でアクセスした場合、データソースに対しては、静的 SQL は実行できません。 DYNAMICRULES オプション (BIND に設定) を、ニックネームを含むパッケージで使用しないでください。

DB2 UDB は DB2 ファミリーのデータ・ソースを使って通信するときに動的 SQL を使用するため、ニックネームを含むパッケージが付加的な許可ステップを必要とする可能性があります。データ・ソースでパッケージを実行する許可 ID には、そのデータ・ソースでパッケージを動的に実行するための適切な権限が必要です。

関連概念:

- 265 ページの『パッケージ経由の間接特権』

ビューを使用したデータ・アクセスの制御

ビューを使用すれば、次のようにして、表に対するアクセス制御や特権付与を行うことができます。

- 表内の指定した列だけにアクセスを制限する

  表内の特定の列だけに対するアクセスが必要なユーザーやアプリケーション・プログラムのために、許可されたユーザーは、必要な列だけを指定したビューを作成できます。
- 表内の行のサブセットだけにアクセスを制限する

  許可されたユーザーは、ビュー定義の副照会の中に WHERE 文節を指定することにより、ビューによってアクセスする行を限定できます。
- データ・ソース表またはビュー内の行のサブセットだけにアクセスを制限します。ニックネームによってデータ・ソースにアクセスしている場合、ニックネームを参照するローカル DB2® Universal Database (DB2 UDB) ビューを作成できます。これらのビューは、1 つまたは複数のデータ・ソースからニックネームを参照することができます。

  注: 複数のデータ・ソースを参照するニックネームを含むビューを作成できるので、ユーザーは 1 つのビューから複数のデータ・ソースのデータにアクセスできます。これらのビューは、*マルチ・ロケーション・ビュー* と呼ばれます。このようなビューは、分散環境全体で重要な表の列の情報を結合する場合や、個々のユーザーに、特定のオブジェクトに対してデータ・ソースで必要な特権がない場合に役立ちます。

ビューを作成するには、SYSADM 権限、DBADM 権限が必要です。または、そのビュー定義の中で参照されるそれぞれの表、ビュー、またはニックネームに対する CONTROL 特権あるいは SELECT 特権が必要です。さらに、ユーザーは、そのビュー用に指定されたスキーマ内にオブジェクトを作成できなければなりません。つ

まり、スキーマがまだ存在していなければ、既存のスキーマに対する CREATEIN 特権、または、データベースに対する IMPLICIT_SCHEMA 権限が必要です。

ニックネームを参照するビューを作成するとき、ビューでニックネームが参照するデータ・ソース・オブジェクト (表とビュー) に対する付加的な権限は必要ありません。ただし、ビューのユーザーがビューにアクセスするとき、基礎となるデータ・ソース・オブジェクトに対する SELECT 権限または同等の権限レベルが必要です。

ユーザーに、基礎となるオブジェクト (表およびビュー) に対してデータ・ソースでの適切な権限がない場合、次のことを実行できます。

1. データ・ソース表の中のユーザーのアクセスを許可する列に対してデータ・ソース・ビューを作成する
2. このビューの SELECT 特権をユーザーに付与する
3. ビューを参照するためのニックネームを作成する

その後、新しいニックネームを参照する SELECT ステートメントを発行することによって、列にアクセスすることができます。

以下のシナリオは、情報へのアクセスを制限するために、ビューを使用する方法をより詳細に示した例です。

次のようなさまざまな理由から、 STAFF 表の情報にアクセスする必要のある人が多数いるとします。たとえば、以下のとおりです。

- 人事部では、表全体についての更新と参照ができなければなりません。

  この要件を満たすのに必要なことは、次のようにして、 PERSONNL グループに STAFF 表に対する SELECT 特権と UPDATE 特権を付与するだけです。

  ```
  GRANT SELECT,UPDATE ON TABLE STAFF TO GROUP PERSONNL
  ```

- 各部署のマネージャーは、部下の給与についての情報を参照する必要があります。

  これは、各部署のマネージャーごとに、専用のビューを作成することによって解決できます。たとえば、51 番の部署のマネージャーに対しては、次のようにビューを作成できます。

  ```
  CREATE VIEW EMP051 AS
     SELECT NAME,SALARY,JOB FROM STAFF
     WHERE DEPT=51
  GRANT SELECT ON TABLE EMP051 TO JANE
  ```

  JANE という許可名を持つマネージャーは、STAFF 表と同じように EMP051 ビューを照会します。このマネージャーが STAFF 表の EMP051 というビューにアクセスするとき、表示される情報は次のようになります。

NAME	SALARY	JOB
Fraye	45150.0	Mgr
Williams	37156.5	Sales
Smith	35654.5	Sales
Lundquist	26369.8	Clerk
Wheeler	22460.0	Clerk

- すべてのユーザーは他の従業員の場所を知る必要があります。この要件を満たすには、次のようにして、 STAFF 表の NAME 列と ORG 表の LOCATION 列についてのビューを作成し、それぞれ DEPT 列と DEPTNUMB 列に基づいて 2 つの表を結合します。

```
CREATE VIEW EMPLOCS AS
   SELECT NAME, LOCATION FROM STAFF, ORG
   WHERE STAFF.DEPT=ORG.DEPTNUMB
GRANT SELECT ON TABLE EMPLOCS TO PUBLIC
```

  従業員の場所に関するビューにアクセスするユーザーは、次の情報を見ることになります。

NAME	LOCATION
Molinare	New York
Lu	New York
Daniels	New York
Jones	New York
Hanes	Boston
Rothman	Boston
Ngan	Boston
Kermisch	Boston
Sanders	Washington
Pernal	Washington
James	Washington
Sneider	Washington
Marenghi	Atlanta
O'Brien	Atlanta
Quigley	Atlanta
Naughton	Atlanta
Abrahams	Atlanta
Koonitz	Chicago
Plotz	Chicago
Yamaguchi	Chicago
Scoutten	Chicago
Fraye	Dallas
Williams	Dallas
Smith	Dallas
Lundquist	Dallas
Wheeler	Dallas
Lea	San Francisco
Wilson	San Francisco
Graham	San Francisco
Gonzales	San Francisco
Burke	San Francisco
Quill	Denver

NAME	LOCATION
Davis	Denver
Edwards	Denver
Gafney	Denver

関連タスク:

- 133 ページの『ビューの作成』
- 261 ページの『特権の付与』

監査機能を使用したデータ・アクセスのモニター

DB2® Universal Database (DB2 UDB) 監査機能は、事前に定義したデータベースのイベントに対して監査履歴を生成し、かつその監査履歴を保存できるようにします。監査機能はデータへのアクセスを妨げる機能ではありませんが、データ・オブジェクトへのアクセス試行または変更試行の記録をモニターして保持することができます。

SYSADM 権限は、監査機能の管理者ツール **db2audit** を使用するのに必須です。

関連概念:

- 281 ページの『DB2 Universal Database (DB2 UDB) 監査機能の紹介』

データ暗号化

セキュリティー計画の一環として、データの暗号化があります。これを行うには、暗号化および暗号化解除組み込み関数である ENCRYPT、DECRYPT_BIN、DECRYPT_CHAR、および GETHINT を使用することができます。

ENCRYPT 関数は、パスワード・ベースの暗号化方式によってデータを暗号化します。これらの関数によって、パスワード・ヒントをカプセル化することもできます。パスワード・ヒントが、暗号化データに組み込まれます。暗号化したデータの暗号化を解除するには、正しいパスワードを使用する必要があります。これらの関数を使用する開発者は、パスワードを忘れた場合の管理や使用できないデータの管理について考慮しなければなりません。

ENCRYPT 関数の結果は、VARCHAR FOR BIT DATA です (32 631 という制限があります)。

暗号化できるデータは、CHAR、VARCHAR、および FOR BIT DATA だけです。

DECRYPT_BIN および DECRYPT_CHAR 関数は、パスワード・ベースの暗号化解除を使用して、データの暗号化を解除します。

DECRYPT_BIN は常に VARCHAR FOR BIT DATA を戻し、 DECRYPT_CHAR は常に VARCHAR を戻します。最初の引き数が CHAR FOR BIT DATA または VARCHAR FOR BIT DATA の可能性があるため、結果が最初の引き数と異なる場合もあります。

結果の長さは、次の 8 バイト境界までのバイト数に依存します。オプションのヒント・パラメーターが指定されている場合、結果の長さは、データ引き数の長さプラス 40 に、次の 8 バイト境界までのバイト数を加えた長さになる可能性があります。オプションのヒント・パラメーターが指定されない場合、結果の長さは、データ引き数の長さプラス 8 に、次の 8 バイト境界までのバイト数を加えた長さになる可能性があります。

GETHINT 関数は、カプセル化されたパスワード・ヒントを返します。パスワード・ヒントとは、データ所有者がパスワードを思い浮かべるために役立つフレーズです。たとえば、パスワード ″Pacific″ を思い浮かべるために、ワード「Ocean」をヒントとして使用することができます。

データの暗号化に使用するパスワードは、以下のいずれかの方法で決定されます。

- パスワード引き数。パスワードは、ENCRYPT 関数が呼び出されるときに明示的に渡されるストリングです。データは、与えられたパスワードで暗号化および暗号化解除されます。
- 暗号化パスワード特殊レジスター。 SET ENCRYPTION PASSWORD ステートメントはパスワード値を暗号化し、その暗号化されたパスワードをデータベース・マネージャーに送信して、特殊レジスターに保管します。パスワード・パラメーターなしで呼び出された ENCRYPT、DECRYPT_BIN、および DECRYPT_CHAR 関数は、 ENCRYPTION PASSWORD 特殊レジスターの値を使用します。 ENCRYPTION PASSWORD 特殊レジスターは、暗号化形式でのみ保管されます。

  特殊レジスターの初期値 (デフォルト値) は空ストリングです。

パスワードに有効な長さは 6 ～ 127 文字です。ヒントに有効な長さは 0 ～ 32 文字です。

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『SET ENCRYPTION PASSWORD ステートメント』
- 「*SQL リファレンス 第 1 巻*」の『DECRYPT_BIN および DECRYPT_CHAR スカラー関数』
- 「*SQL リファレンス 第 1 巻*」の『ENCRYPT スカラー関数』
- 「*SQL リファレンス 第 1 巻*」の『GETHINT スカラー関数』

タスクおよび実行に必要な許可

仕事の責任の分担方法は、それぞれの組織によって違います。 表 6 に、他の一般的な仕事の種類、それらの仕事に通常伴うタスク、およびそれらのタスクを行うために必要な権限または特権を挙げます。

表 6. 一般的な仕事の種類、タスク、および必要な権限許可

仕事の種類	タスク	必要な許可
部署管理者	部署のシステムを監督する。データベースを作成する。	SYSCTRL 権限。部署に独自のインスタンスのある場合は SYSADM 権限。

表 6. 一般的な仕事の種類、タスク、および必要な権限許可 (続き)

仕事の種類	タスク	必要な許可
セキュリティー管理者	権限許可と特権の一部または全部を他のユーザーに許可する。	SYSADM または DBADM 権限。
データベース管理者	データベースの設計、開発、操作、保全、保守を行う。	データベースについての DBADM および SYSMAINT 権限。場合によっては、SYSCTRL 権限。
システム・オペレーター	データベースをモニターし、バックアップ機能を実行する。	SYSMAINT 権限。
アプリケーション・プログラマー	データベース・マネージャーのアプリケーション・プログラムの開発およびテストを行う。テスト・データの表を作成することもある。	既存のパッケージに対する BINDADD、BIND 特権、 1 つ以上のデータベースの CONNECT および CREATETAB 特権、一部の特定のスキーマの特権、および一部の表についての一連の特権。 CREATE_EXTERNAL_ROUTINE が必要な場合もあります。
ユーザー・アナリスト	システム・カタログ・ビューを調べて、アプリケーション・プログラムのデータ要件を定義する。	カタログ・ビューの SELECT 特権。 1 つ以上のデータベースの CONNECT 特権。
プログラム・エンド・ユーザー	アプリケーション・プログラムを実行する。	パッケージの EXECUTE 特権。 1 つ以上のデータベースの CONNECT 特権。この表の下にある注記を参照してください。
インフォメーション・センター・コンサルタント	照会ユーザーについてのデータ要件を定義する。表とビューを作成し、データベース・オブジェクトへのアクセス権を付与することによって、データを提供する。	データベースについての DBADM 権限。
照会ユーザー	データの検索、追加、削除、または変更のために SQL ステートメントを出す。場合によっては、結果を表に保管する。	1 つ以上のデータベースの CONNECT 特権。作成する表およびビューのスキーマの CREATEIN 特権。一部の表およびビューの SELECT、 INSERT、 UPDATE、 DELETE 特権。

注: アプリケーション・プログラムに動的 SQL ステートメントが含まれている場合、プログラムのエンド・ユーザーには、 EXECUTE と CONNECT に加えてさらにほかの特権 (SELECT、INSERT、DELETE、および UPDATE など) が必要になる場合があります。

関連概念:

- 248 ページの『システム管理権限 (SYSADM)』
- 248 ページの『システム制御権限 (SYSCTRL)』
- 249 ページの『システム保守権限 (SYSMAINT)』
- 250 ページの『データベース管理権限 (DBADM)』
- 252 ページの『LOAD 権限』

- 252 ページの『データベース権限』

関連タスク:

- 261 ページの『特権の付与』
- 262 ページの『特権の取り消し』

セキュリティー強化のためのシステム・カタログの使用

それぞれのデータベースについての情報は、 (データベース生成時に作成される) システム・カタログという 1 組のビュー内に自動的に維持されます。このシステム・カタログには、表、列、索引、プログラム、特権、その他のオブジェクトが含まれています。

これらのビューには、ユーザーによって保持される特権と、それぞれの特権を付与するユーザーのアイデンティティーがリストされています。

SYSCAT.DBAUTH	データベースの特権のリスト
SYSCAT.TABAUTH	表とビューの特権のリスト
SYSCAT.COLAUTH	列の特権のリスト
SYSCAT.PACKAGEAUTH	パッケージの特権のリスト
SYSCAT.INDEXAUTH	索引の特権のリスト
SYSCAT.SCHEMAAUTH	スキーマの特権のリスト
SYSCAT.PASSTHRUAUTH	サーバーの特権のリスト
SYSCAT.ROUTINEAUTH	ルーチン (関数、メソッド、およびプロシージャー) の特権のリスト

システムによってユーザーに付与される特権の付与者は、SYSIBM になります。SYSADM、SYSMAINT および SYSCTRL はシステム・カタログにリストされます。

CREATE ステートメントと GRANT ステートメントを使うと、システム・カタログの中に特権が入れられます。 SYSADM および DBADM 権限を持つユーザーは、システム・カタログ・ビューに対する SELECT 特権の GRANT と取り消しを行うことができます。

関連タスク:

- 274 ページの『付与された特権を持つ許可名の検索』
- 274 ページの『DBADM 権限を持つすべての名前の検索』
- 275 ページの『表へのアクセスを許可されている名前の検索』
- 275 ページの『ユーザーに付与されたすべての特権の検索』
- 276 ページの『システム・カタログ・ビューのセキュリティー』

関連資料:

- 「*SQL リファレンス 第 1 巻*」の『SYSCAT.COLAUTH カタログ・ビュー』
- 「*SQL リファレンス 第 1 巻*」の『SYSCAT.DBAUTH カタログ・ビュー』
- 「*SQL リファレンス 第 1 巻*」の『SYSCAT.INDEXAUTH カタログ・ビュー』

- 「*SQL リファレンス 第 1 巻*」の『SYSCAT.PACKAGEAUTH カタログ・ビュー』
- 「*SQL リファレンス 第 1 巻*」の『SYSCAT.SCHEMAAUTH カタログ・ビュー』
- 「*SQL リファレンス 第 1 巻*」の『SYSCAT.TABAUTH カタログ・ビュー』
- 「*SQL リファレンス 第 1 巻*」の『SYSCAT.PASSTHRUAUTH カタログ・ビュー』
- 「*SQL リファレンス 第 1 巻*」の『SYSCAT.ROUTINEAUTH カタログ・ビュー』

セキュリティー強化のためのシステム・カタログの使用に関する詳細

このセクションでは、データベース内での特権の所有者とその種類を判別するいくつかの方法について検討します。

付与された特権を持つ許可名の検索

手順:

すべての特権に関する情報が、いずれか 1 つのシステム・カタログ・ビューに含まれることはありません。以下のステートメントは、特権を持つすべての許可名を検索します。

```
SELECT DISTINCT GRANTEE, GRANTEETYPE, 'DATABASE' FROM SYSCAT.DBAUTH
UNION
SELECT DISTINCT GRANTEE, GRANTEETYPE, 'TABLE   ' FROM SYSCAT.TABAUTH
UNION
SELECT DISTINCT GRANTEE, GRANTEETYPE, 'PACKAGE ' FROM SYSCAT.PACKAGEAUTH
UNION
SELECT DISTINCT GRANTEE, GRANTEETYPE, 'INDEX   ' FROM SYSCAT.INDEXAUTH
UNION
SELECT DISTINCT GRANTEE, GRANTEETYPE, 'COLUMN  ' FROM SYSCAT.COLAUTH
UNION
SELECT DISTINCT GRANTEE, GRANTEETYPE, 'SCHEMA  ' FROM SYSCAT.SCHEMAAUTH
UNION
SELECT DISTINCT GRANTEE, GRANTEETYPE, 'SERVER  ' FROM SYSCAT.PASSTHRUAUTH
ORDER BY GRANTEE, GRANTEETYPE, 3
```

時折、このステートメントによって検索されたリストと、システムのセキュリティー機能で定義されているユーザー名とグループ名のリストとを比較してみる必要があります。これによって、有効でなくなった許可名を識別できます。

注: リモート・データベース・クライアントをサポートする場合、許可名をリモート・クライアントだけに定義し、データベースのサーバー・マシンには定義しないことも可能です。

関連概念:

- 273 ページの『セキュリティー強化のためのシステム・カタログの使用』

DBADM 権限を持つすべての名前の検索

手順:

以下のステートメントは、DBADM 権限が直接付与されている、すべての許可名を検索します。

```
SELECT DISTINCT GRANTEE FROM SYSCAT.DBAUTH
   WHERE DBADMAUTH = 'Y'
```

関連概念:

- 250 ページの『データベース管理権限 (DBADM)』
- 273 ページの『セキュリティー強化のためのシステム・カタログの使用』

表へのアクセスを許可されている名前の検索

手順:

以下のステートメントは、修飾子 JAMES を持つ表 EMPLOYEE にアクセスすることが直接許可されている、すべての許可名を検索します。

```
SELECT DISTINCT GRANTEETYPE, GRANTEE FROM SYSCAT.TABAUTH
   WHERE TABNAME = 'EMPLOYEE'
     AND TABSCHEMA = 'JAMES'
UNION
SELECT DISTINCT GRANTEETYPE, GRANTEE FROM SYSCAT.COLAUTH
   WHERE TABNAME = 'EMPLOYEE'
     AND TABSCHEMA = 'JAMES'
```

だれが修飾子 JAMES を持つ表 EMPLOYEE を更新できるかを調べるためには、以下のステートメントを出します。

```
SELECT DISTINCT GRANTEETYPE, GRANTEE FROM SYSCAT.TABAUTH
   WHERE TABNAME = 'EMPLOYEE' AND TABSCHEMA = 'JAMES' AND
      (CONTROLAUTH  = 'Y' OR
       UPDATEAUTH   = 'Y' OR  UPDATEAUTH   = 'G')
UNION
SELECT DISTINCT GRANTEETYPE, GRANTEE FROM SYSCAT.DBAUTH
   WHERE DBADMAUTH = 'Y'
UNION
SELECT DISTINCT GRANTEETYPE, GRANTEE FROM SYSCAT.COLAUTH
   WHERE TABNAME = 'EMPLOYEE' AND TABSCHEMA = 'JAMES' AND
   PRIVTYPE = 'U'
```

これは、DBADM 権限をもつ許可名があればすべて検索し、さらに CONTROL または UPDATE 特権が直接付与されている許可名も検索します。ただし、SYSADM 権限だけを保持しているユーザーの許可名は戻しません。

一部の許可名は、個別のユーザーだけでなく、グループである場合もあることに注意してください。

関連概念:

- 256 ページの『表およびビューの特権』
- 273 ページの『セキュリティー強化のためのシステム・カタログの使用』

ユーザーに付与されたすべての特権の検索

手順:

ユーザーは、システム・カタログ・ビューについての照会を行うことにより、自ら持っている特権のリストと、他のユーザーに付与した特権のリストを作成できます。たとえば、以下のステートメントは、個々の許可名に直接付与されているデータベース特権のリストを検索します。

```
SELECT * FROM SYSCAT.DBAUTH
   WHERE GRANTEE = USER AND GRANTEETYPE = 'U'
```

表の特権のうち特定のユーザーによって直接付与されたものを検索するには、次のようなステートメントを使います。

```
SELECT * FROM SYSCAT.TABAUTH
   WHERE GRANTOR  = USER
```

以下のステートメントは、特定のユーザーによって直接付与された、個別の列特権のリストを検索します。

```
SELECT * FROM SYSCAT.COLAUTH
   WHERE GRANTOR  = USER
```

このステートメントの中の USER というキーワードは、常にユーザーの許可名の値と等しくなります。 USER は読み取り専用の特殊レジスターです。

関連概念:

- 242 ページの『特権、権限レベル、およびデータベース権限』
- 252 ページの『データベース権限』
- 273 ページの『セキュリティー強化のためのシステム・カタログの使用』

関連タスク:

- 261 ページの『特権の付与』
- 262 ページの『特権の取り消し』

システム・カタログ・ビューのセキュリティー

手順:

データベースの生成時に、システム・カタログ・ビューに対する SELECT 特権が PUBLIC に付与されます。多くの場合、これによってセキュリティー上の問題が生じることはありません。しかし、これらの表にはデータベース内のすべてのオブジェクトが含まれているため、非常に重要なデータの場合は適切でないことがあります。このような場合には、PUBLIC から SELECT 特権を取り消してから、必要に応じて、特定のユーザーに対して SELECT 特権を付与することを考慮してください。システム・カタログ・ビューについての SELECT 特権の付与と取り消しは、他のビューの場合と同じ方法で行いますが、そのためには SYSADM または DBADM 権限が必要です。

少なくとも、次のカタログ・ビューに対するアクセスを制限することを考慮すべきです。

- SYSCAT.DBAUTH
- SYSCAT.TABAUTH
- SYSCAT.PACKAGEAUTH
- SYSCAT.INDEXAUTH

- SYSCAT.COLAUTH
- SYSCAT.PASSTHRUAUTH
- SYSCAT.SCHEMAAUTH

それによって、ユーザー特権についての情報がデータベースにアクセスできる人全員で利用できるような事態を避けることができます。この情報を使用して、不適切なユーザーがデータベースに対する無許可アクセスを取得する可能性があります。

各列にどの統計が集められているかも調べてください。システム・カタログに記録される統計には、ご使用の環境では機密情報となりうるデータ値が含まれることがあります。この統計に機密データが含まれている場合には、 SYSCAT.COLUMNS および SYSCAT.COLDIST カタログ・ビューについての SELECT 特権を、PUBLIC から取り消すことができます。

システム・カタログ・ビューに対するアクセスを限定したい場合は、それぞれの許可名が自分自身の特権にかかわる情報だけを検索できるようにするビューを定義することができます。

たとえば、次のビュー MYSELECTS には、ユーザーの許可名に SELECT 特権が直接付与されている表の所有者と名前が含まれます。

```
CREATE VIEW MYSELECTS AS
      SELECT TABSCHEMA, TABNAME FROM SYSCAT.TABAUTH
      WHERE GRANTEETYPE = 'U'
        AND GRANTEE = USER
        AND SELECTAUTH = 'Y'
```

このステートメントの中の USER というキーワードは、常に、許可名の値と等しくなります。

以下のステートメントは、このビューをそれぞれの許可名から利用可能にするものです。

```
GRANT SELECT ON TABLE MYSELECTS TO PUBLIC
```

最後に、基本表についての SELECT 特権を必ず取り消すようにしてください。

```
REVOKE SELECT ON TABLE SYSCAT.TABAUTH FROM PUBLIC
```

関連概念:

- 「*管理ガイド: パフォーマンス*」の『カタログ統計』
- 252 ページの『データベース権限』
- 273 ページの『セキュリティー強化のためのシステム・カタログの使用』

関連タスク:

- 261 ページの『特権の付与』
- 262 ページの『特権の取り消し』

ファイアウォール・サポートの紹介

ファイアウォール は、ネットワーク・ゲートウェイ・サーバーに位置し、システムまたはネットワークへの無許可アクセスを防ぐために使用される、関連したプログラムのセットです。

ファイアウォールには、以下の 4 つのタイプがあります。

1. ネットワーク・レベル、パケット・フィルター、またはスクリーニング・ルーター・ファイアウォール
2. 従来のアプリケーション・レベルのプロキシー・ファイアウォール
3. 回線レベル、または透過性のプロキシー・ファイアウォール
4. Stateful Multi-Layer Inspection (SMLI) ファイアウォール

上記のファイアウォールのタイプのいずれかに入る既存のファイアウォール製品もあります。他にも、上記のタイプの組み合わせとなる、多くのファイアウォール製品があります。

関連概念:

- 278 ページの『スクリーニング・ルーター・ファイアウォール』
- 279 ページの『アプリケーション・プロキシー・ファイアウォール』
- 279 ページの『回線レベルのファイアウォール』
- 279 ページの『Stateful Multi-Layer Inspection (SMLI) ファイアウォール』

スクリーニング・ルーター・ファイアウォール

このタイプのファイアウォールはネットワーク・レベルまたはパケット・フィルター・ファイアウォールとも呼ばれます。このようなファイアウォールは、入ってくるパケットをプロトコル属性によってスクリーニングすることにより動作します。スクリーニングされるプロトコル属性には、送信元または宛先アドレス、プロトコルのタイプ、送信元または宛先のポート、または他のプロトコルに特有の属性が含まれます。

すべてのファイアウォール・ソリューション (SOCKS を除く) では、 DB2® Universal Database (DB2 UDB) によって使用されるすべてのポートを出入りするパケットのために開けておく必要があります。 DB2 UDB は、DB2 UDB ツールによって使用される DB2 Administration Server (DAS) 用として、ポート 523 を使用します。サービス・ファイルを使用して、サーバー・データベース・マネージャー構成ファイル内のサービス名をそのポート番号にマップし、すべてのサーバー・インスタンスが使用するポートを判別してください。

関連概念:

- 278 ページの『ファイアウォール・サポートの紹介』

アプリケーション・プロキシー・ファイアウォール

プロキシーまたはプロキシー・サーバーは、 Web クライアントと Web サーバーの間の中継地点として動作する技術です。プロキシー型ファイアウォールは、クライアントからの要求に対するゲートウェイの役割を果たします。クライアントの要求がファイアウォールで受信されると、最終のサーバー宛先アドレスがプロキシー・ソフトウェアによって判別されます。アプリケーション・プロキシーがアドレスを変換し、必要に応じてさらにアクセス・コントロール検査およびロギングを行い、クライアントに代わってサーバーに接続します。

ファイアウォール・マシン上の DB2® Connect 製品は、宛先サーバーに対するプロキシーの役割を果たすことができます。また、最終的な宛先サーバーに対するホップ・サーバーとして動作する、ファイアウォール上の DB2 Universal Database™ (DB2 UDB) サーバーは、アプリケーション・プロキシーに似た役割を果たします。

関連概念:

- 278 ページの『ファイアウォール・サポートの紹介』

回線レベルのファイアウォール

このタイプのファイアウォールは、透過性プロキシー・ファイアウォールとも呼ばれます。透過性プロキシー・ファイアウォールは、プロキシー認証および識別に必要とされる以上には、要求または応答を変更しません。透過性プロキシー・ファイアウォールとしては、 SOCKS があります。

DB2® Universal Database (DB2 UDB) は、SOCKS バージョン 4 をサポートします。

関連概念:

- 278 ページの『ファイアウォール・サポートの紹介』

Stateful Multi-Layer Inspection (SMLI) ファイアウォール

このタイプのファイアウォールは、オープン・システム間相互接続 (OSI) モデルの 7 層すべてを調べる、パケット・フィルター操作の高性能な形式です。各パケットが調べられ、類似したパケットの既知の状態と比較されます。スクリーニング・ルーター・ファイアウォールがパケット・ヘッダーのみを調べるのに対し、 SMLI ファイアウォールはデータも含むパケット全体を調べます。

関連概念:

- 278 ページの『ファイアウォール・サポートの紹介』

第 8 章 DB2 Universal Database™ (DB2 UDB) アクティビティーの監査

DB2 Universal Database (DB2 UDB) 監査機能の紹介

認証、権限、および、特権はデータへの既知のまたは予期されたアクセスを制御するために使用できますが、これらの方法はデータへの未知のまたは予期されないアクセスを防ぐには十分ではない可能性があります。後者のタイプのデータ・アクセスの検出を支援するために、 DB2® Universal Database (DB2 UDB) には監査機能があります。不適切なデータ・アクセスを正常にモニターして分析することにより、データ・アクセスの制御を改善し、データへの悪意のあるまたは不注意な無許可アクセスを最終的に防止することができます。システム管理処置を含む、アプリケーションおよび個々のユーザー・アクセスのモニターは、データベース・システムのアクティビティーの履歴レコードを提供します。

DB2 UDB 監査機能は、事前に定義したデータベースのイベントに対して監査履歴を生成し、かつその監査履歴を保存できるようにします。この機能により生成されたレコードは、監査ログ・ファイルに保持されます。これらのレコードを分析することで、使用パターンが明らかになり、システム誤用を識別することができます。そのようなシステム誤用を識別した場合、それを制限または除去できます。

監査機能はインスタンス・レベルで作用し、すべてのインスタンス・レベルのアクティビティーおよびデータベース・レベルのアクティビティーを記録します。

パーティション・データベース環境において作動しているとき、ユーザーが接続しているパーティション (コーディネーター・ノード)、またはカタログ・ノード (同じパーティションではない場合) で、多数の監査可能なイベントが発生します。つまり、監査レコードが複数のパーティションによって生成される可能性があります。各監査レコードの一部は、コーディネーター・ノードおよび起点ノード ID に関する情報を含んでいます。

監査ログ (db2audit.log) および監査構成ファイル (db2audit.cfg) は、インスタンスの `security` サブディレクトリーに置かれます。インスタンスを作成するとき、可能な場合には、読み取り/書き込み許可がオペレーティング・システムによってこれらのファイルに設定されます。デフォルトでは、この許可はインスタンス所有者専用の読み取り/書き込みとなります。これらの許可を変更しないようお勧めします。

監査機能の管理者ツール (db2audit) のユーザーは、 SYSADM 権限を持っていなければなりません。

監査機能は、明示的に停止および開始される必要があります。開始するとき、監査機能は既存の監査構成情報を使用します。監査機能は DB2 UDB サーバーから独立しているので、インスタンスが停止されてもアクティブのままです。実際、インスタンスが停止されるとき、監査レコードが監査ログに生成される可能性があります。

監査機能の許可ユーザーは、監査機能の中で次の動作を制御できます。

- DB2 UDB インスタンスの中で監査可能なイベントの記録を開始する。
- DB2 UDB インスタンスの中で監査可能なイベントの記録を停止する。
- 記録される監査可能なイベントの区分の選択を含め、監査機能の動作を構成する。
- 現在の監査構成の記述を要求する。
- インスタンスからすべてのペンディングになっている監査レコードをフラッシュし、それらを監査ログに書き込む。
- 監査ログから監査レコードをフラット・ファイルまたは ASCII 区切りファイルにフォーマットしてコピーすることにより、監査レコードを抽出する。抽出は、ログ・レコード分析の準備、またはログ・レコードの整理の準備のいずれかの理由で行われます。
- 現行監査ログにある監査レコードを整理する。

注: 監査ユーティリティーを使用する前に、 db2audit start コマンドを実行して監査機能がオンになっていることを確認してください。

生成される監査レコードには様々な区分があります。監査するために使用可能なイベントの区分の記述 (下記参照) において、各区分の名前に続いて、区分タイプの識別に使用される 1 つの単語のキーワードがあることに注意してください。監査のために使用可能なイベントの区分は次のようになります。

- 監査 (AUDIT)。監査設定が変更されるとき、または監査ログがアクセスされるときにレコードを生成する。
- 許可検査 (CHECKING)。 DB2 UDB オブジェクトや関数に対するアクセス試行または操作試行の許可検査のときに、レコードを生成する。
- オブジェクト保守 (OBJMAINT)。データ・オブジェクトを作成またはドロップするときにレコードを生成する。
- セキュリティー保守 (SECMAINT)。オブジェクトまたはデータベース特権、つまり DBADM 権限を付与あるいは取り消すとき、レコードを生成する。 データベース・マネージャーのセキュリティー構成パラメーター SYSADM_GROUP、SYSCTRL_GROUP、または SYSMAINT_GROUP が変更されるときにもまたレコードは生成される。
- システム管理 (SYSADMIN)。 SYSADM、SYSMAINT、または SYSCTRL 権限を必要とする操作が実行されるとき、レコードを生成する。
- ユーザー検証 (VALIDATE)。ユーザーを認証しているとき、またはシステムのセキュリティー情報を検索しているときにレコードを生成する。
- 操作コンテキスト (CONTEXT)。データベースの操作が実行されるとき、操作コンテキストを表示するレコードを生成する。この区分を使用すると、監査ログ・ファイルのより良い変換処理を可能にします。ログのイベント相関関係子フィールドを同時に使用することで、イベントのグループを 1 つのデータベース操作に戻って関連付けることができます。たとえば、動的 SQL の SQL ステートメント、静的 SQL のパッケージ ID、つまり CONNECT のような実行されている操作タイプのインディケーターは、監査結果を分析しているときに必要なコンテキストを提供できます。

注: 操作コンテキストを提供する SQL ステートメントは、かなり長くなる可能性があり、コンテキストのレコード内にすべてが示されます。このため、CONTEXT レコードが非常に大きくなる可能性があります。

- 失敗、成功、またはその両方を監査できます。

すべてのデータベース操作に対して、複数のレコードが生成される可能性があります。監査ログに生成かつ移動されるレコードの実際の数は、監査機能構成により指定された、記録されるイベントの区分の数によって決定されます。さらに、成功、失敗、またはその両方を監査するかどうかによっても異なります。このため、監査対象のイベントの選択は重要です。

関連概念:

- 283 ページの『監査機能の動作』
- 298 ページの『監査機能のレコード設計の紹介』
- 315 ページの『監査機能のヒントと技法』

関連タスク:

- 317 ページの『DB2 UDB 監査機能アクティビティーの制御』

関連資料:

- 285 ページの『監査機能の使用方法』
- 297 ページの『監査機能のメッセージ』

監査機能の動作

監査機能は、データベース・インスタンスに影響を及ぼすイベントを含む監査可能なイベントを記録します。このため、監査機能は DB2® Universal Database (DB2 UDB) の独立した部分であり、 DB2 UDB のインスタンスが停止されても動作可能です。監査機能がアクティブである場合、停止されたインスタンスが開始されるときに、インスタンス内のデータベース・イベントに対する監査が再開します。

監査ログへの監査レコードの書き込みタイミングは、インスタンスのデータベースのパフォーマンスに大きな影響を与えます。監査レコードの書き込みは、そのレコードの生成の原因となるイベントの発生と同期的に、または非同期的に行われます。 *audit_buf_sz* データベース・マネージャーの構成パラメーター値は、いつ監査レコードが書き込まれるかを決定します。

このパラメーターの値がゼロ (0) であれば、書き込みは同期的に行われます。監査レコードを生成しているイベントは、そのレコードがディスクに書き込まれるまで待機します。それぞれのレコードに関連した待機のために、DB2 UDB のパフォーマンスが低下します。

audit_buf_sz の値がゼロより大きい場合、レコードの書き込みは非同期で行われます。ゼロより大きいとき *audit_buf_sz* の値は、内部バッファーの作成に使用される 4 KB ページの数となります。ディスクに書き込む前の複数の監査レコードを保持するために、内部バッファーが使用されます。監査イベントの結果として監査レコードを生成するステートメントは、レコードがディスクに書き込まれるまで待機することなく、動作を継続できます。

非同期の場合、空きがあるバッファーに監査レコードがしばらく保持される可能性があります。長時間にわたってこれが発生しないようにするために、データベース・マネージャーは定期的に監査レコードの書き込みを強制します。さらに、監査機能の許可ユーザーもまた、明示的な要求により監査バッファーをフラッシュすることができます。

エラーが発生したときには、同期のレコード書き込みか、非同期のレコード書き込みかによって、違いが出てきます。非同期モードでは、監査レコードがディスクに書き込まれる前にバッファーに入れられるので、いくつかのレコードが失われる可能性があります。同期モードでは、エラーのために書き込みできない監査レコードは多くて 1 つなので、失われる可能性があるレコードは 1 つだけです。

ERRORTYPE 監査機能パラメーターの設定により、どのように DB2 UDB と監査機能の間でエラーを管理するかを制御します。監査機能がアクティブであり、監査機能パラメーター ERRORTYPE の設定が AUDIT であるとき、監査機能は DB2 UDB の他の部分と同じように扱われます。監査レコードを (同期モードの場合はディスクに、非同期モードの場合は監査バッファーに) 書き込まないと、ステートメントに関連した監査イベントは正常終了したものと見なされません。このモードの実行時にエラーが検出されると、監査レコードを生成するステートメントに関する負の SQLCODE がアプリケーションに戻されます。エラー・タイプが NORMAL に設定されると、 db2audit からのエラーはすべて無視され、操作の SQLCODE が戻されます。

API または SQL ステートメントおよび DB2 UDB インスタンスの監査設定に応じて、特定のイベントに対して監査レコードが何も生成されなかったり、 1 つまたは複数の監査レコードが生成されたりします。たとえば、SELECT 副照会による SQL UPDATE ステートメントの結果として、表の UPDATE 特権に対する許可検査の結果を含む 1 つの監査レコードと、表の SELECT 特権に対する許可検査の結果を含む別の監査レコードが生成される可能性があります。

動的なデータ操作言語 (DML) のステートメントの場合、ステートメントが準備される時点ですべての許可検査の監査レコードが生成されます。同一ユーザーによるステートメントの再使用では許可検査されないため、再度監査されることはありません。ただし、特権情報を含んでいるカタログ表のいずれかが変更された場合には、次の作業単位において、キャッシュされた動的 SQL ステートメントのステートメント特権が再び検査され、 1 つまたは複数の新規監査レコードが作成されます。

静的 DML ステートメントだけを含んでいるパッケージの場合、監査レコードを生成する可能性のある唯一の監査可能イベントは、ユーザーがそのパッケージを実行する特権を持っているかどうかの許可検査です。パッケージ内の静的 SQL ステートメント用に必要な許可検査および監査レコードの作成は、パッケージがプリコンパイルまたはバインドされるときに実行されます。パッケージ内部の静的 SQL ステートメントの実行は、監査できません。ユーザーにより明示的に、またはシステムにより暗黙的に 1 つのパッケージがバインドされるとき、静的 SQL ステートメントにより要求される許可検査のために複数の監査レコードが生成されます。

許可検査が実行時に行われるステートメント (たとえば、データ定義言語 (DDL)、GRANT、および REVOKE ステートメント) の場合、これらのステートメントが使用されるときにはいつでも監査レコードが生成されます。

注: DDL を実行するとき、監査レコード内の (コンテキスト・イベントを除く) すべてのイベント用に記録されるセクション番号は、ステートメントの実際のセクション番号にかかわらずゼロ (0) にリセットされます。

関連概念:

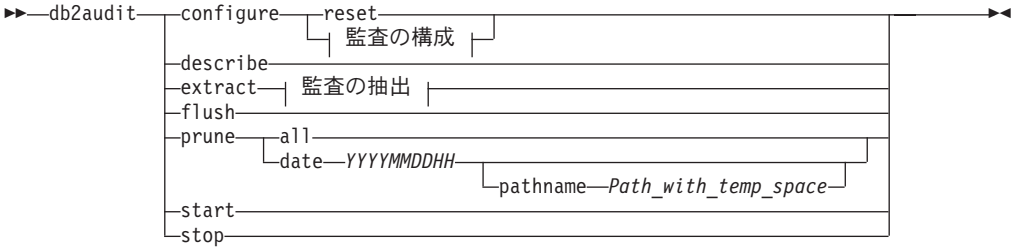
- 281 ページの『DB2 Universal Database (DB2 UDB) 監査機能の紹介』

関連資料:

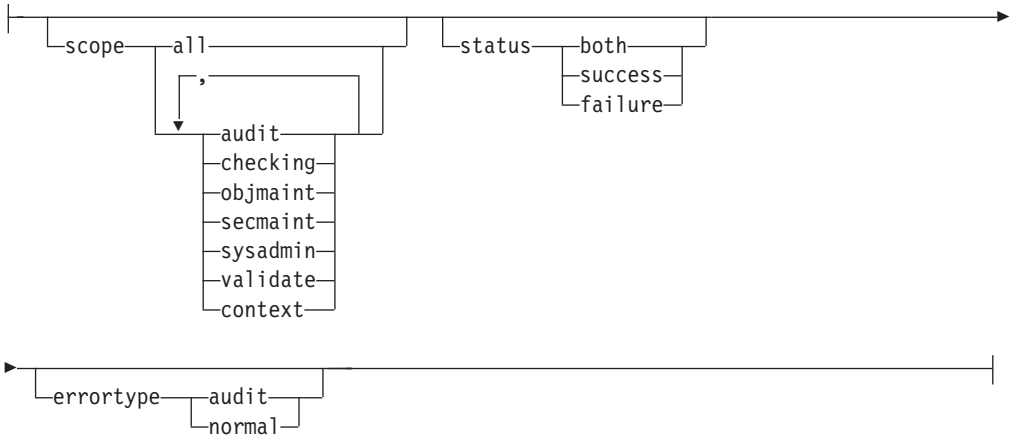
- 「*管理ガイド: パフォーマンス*」の『audit_buf_sz - 「監査バッファー・サイズ」構成パラメーター』
- 285 ページの『監査機能の使用方法』

監査機能の使用方法

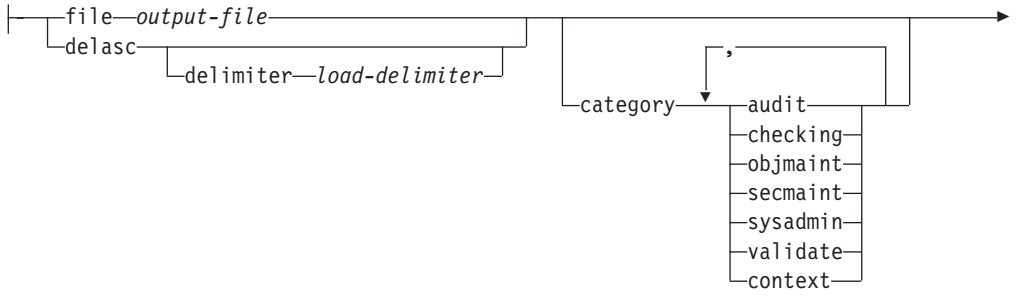
次の構文図の各部分を検討すれば、監査機能の使用方法を理解できます。

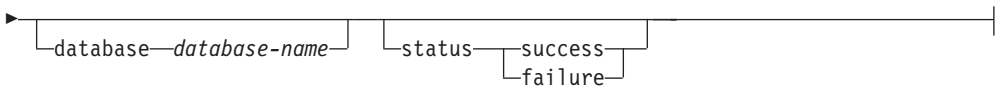


監査の構成:



監査の抽出:





下記に、パラメーターの説明と暗黙の使用を示します。

configure

このパラメーターは、インスタンスの security サブディレクトリーにある db2audit.cfg 構成ファイルの変更を可能にします。このファイルは、インスタンスがシャットダウンしていても更新できます。インスタンスがアクティブなときに更新すると、すべてのパーティションにわたる DB2 Universal Database™ (DB2 UDB) の監査に動的に影響します。監査機能が開始済みで、かつ監査可能イベントの *audit* 区分が監査されているならば、構成ファイルに対する構成操作によって監査レコードが作成されます。

構成ファイルに対する可能な操作は以下のとおりです。

- RESET。この操作によって構成ファイルは初期構成に戻されます (SCOPE は CONTEXT を除くすべての区分を表し、 STATUS は FAILURE、ERRORTYPE は NORMAL、監査機能は OFF)。元の監査構成ファイルが欠落または損傷している場合、この操作によって新しい監査構成ファイルを作成することができます。
- SCOPE。この操作は、監査対象のイベントの区分 (1 つまたは複数) を指定します。さらに、この操作によって特定の監査対象に絞り込み、ログの増大を抑えることができます。ログに記録されるイベントの数とタイプをできるだけ制限することをお勧めします。そうしないと、監査ログは急速に増加します。

  注: デフォルト SCOPE が CONTEXT を除くすべての区分であることに注意してください。その結果、レコードが急速に生成される可能性があります。モード (同期または非同期) および区分の選択によって、パフォーマンスがかなり低下し、ディスク要件が著しく増加する可能性があります。
- STATUS。この操作は、正常終了のイベントまたは失敗したイベントのどちらかだけを記録するか、それとも両方のイベントを記録するかを指定する。

  注: コンテキスト・イベントは、操作の状況が認識される前に発生します。このため、そのようなイベントは、このパラメーターの値に関係なく記録されます。
- ERRORTYPE。この操作は、監査エラーをユーザーに戻すか、それとも無視するかを指定する。このパラメーター値として、以下が可能です。
  - AUDIT。監査機能の内部で発生するエラーを含むすべてのエラーが DB2 UDB によって管理され、負の SQLCODE がすべて呼び出し元に報告される。
  - NORMAL。 db2audit によって生成されたすべてのエラーは無視され、実行されている操作に関連したエラーの SQLCODE だけがアプリケーションに戻される。

describe
このパラメーターは、現在の監査構成情報と状況を標準出力に表示する。

extract
このパラメーターは、監査ログから指示された宛先への監査レコードの移動を可能にする。オプションの文節をまったく指定しない場合、監査レコードのすべてが抽出され、フラット・レポート・ファイルに格納されます。*output_file* がすでに存在する場合、エラー・メッセージが戻されます。

下記に、抽出する際に使用できる可能なオプションを示します。

- FILE。抽出される監査レコードをファイル (*output_file*) に格納する。ファイル名を指定しない場合、レコードは `sqllib` の `security` サブディレクトリーの `db2audit.out` ファイルに書き込まれます。ディレクトリーが指定されない場合、 *output_file* は現行作業ディレクトリーに書き込まれます。
- DELASC。抽出された監査レコードを、 DB2 UDB リレーショナル表にロードするのに適切な区切り文字付き ASCII 形式で格納する。出力は区分ごとに 1 つずつ独立したファイルに置かれます。ファイル名は次のとおりです。
  - `audit.del`
  - `checking.del`
  - `objmaint.del`
  - `secmaint.del`
  - `sysadmin.del`
  - `validate.del`
  - `context.del`

  これらのファイルは、常に `sqllib` の `security` サブディレクトリーに格納されます。

  さらに、DELASC の指定内容によって、監査ログから抽出するとき、デフォルトの監査文字ストリング区切り文字 (『0xff』) をオーバーライドすることができます。監査レコードを保持する表にロードするために使用したい新しい区切り文字を、 DELASC DELIMITER の後に続けます。この新しいロード区切り文字として、単一文字 (たとえば、!) または 16 進数で示される 4 バイト・ストリング (たとえば 0xff) のいずれかが可能です。
- CATEGORY。指定された区分の監査イベントの監査レコードが抽出される。これを指定しない場合、すべての区分が抽出の対象となります。
- DATABASE。指定されたデータベースの監査レコードが抽出される。これを指定しない場合、すべてのデータベースが抽出の対象となります。
- STATUS。指定された状況の監査レコードが抽出される。これを指定しない場合、すべてのレコードが抽出の対象となります。

flush このパラメーターは、すべてのペンディングの監査レコードを監査ログに書き込むよう強制する。さらに、監査機能がエラー状態の場合、監査状態はエンジンにおいて「ログ利用不可」から「ログ作動可能」にリセットされます。

prune このパラメーターを使用すると、監査ログから監査レコードの削除を可能に

する。監査機能がアクティブで、 『audit』 区分のイベントが監査対象として指定されている場合、監査ログが整理された後に監査レコードが記録されます。

下記に、整理に使用できるオプションを示します。

- ALL。監査ログにあるすべての監査レコードが削除される。
- DATE yyyymmddhh。指定された日付/時間に発生した、または指定された日付/時間以前に発生したすべての監査レコードを監査ログから削除するよう指定できる。ユーザーは、オプションで、監査記録を整理するときに監査機能が一時スペースとして使用する

  ```
  pathname
  ```

  を指定できます。この一時スペースを使用すると、監査ログのあるディスク容量がいっぱいで、整理操作に十分なスペースがない場合にも、監査ログを整理できます。

start このパラメーターは、 db2audit.cfg ファイルの内容に基づくイベントの監査を開始させます。パーティション DB2 UDB インスタンスでは、この文節が指定されると、すべてのパーティションに対する監査が開始します。 『audit』 区分のイベントが監査対象として指定されている場合、監査機能が開始されるときに 1 つの監査レコードが記録されます。

stop このパラメーターを使用すると、監査機能によるイベント監査を停止できます。パーティション DB2 UDB インスタンスでは、この文節が指定されると、すべてのパーティションに対する監査が停止します。 『audit』 区分のイベントが監査対象として指定されている場合、監査機能が停止されるときに 1 つの監査レコードが記録されます。

関連概念:

- 281 ページの『DB2 Universal Database (DB2 UDB) 監査機能の紹介』
- 315 ページの『監査機能のヒントと技法』

関連資料:

- 「コマンド・リファレンス」の『db2audit - 監査機能管理者ツール・コマンド』

DB2 表での DB2 監査データの操作

以下のトピックでは、DB2 監査データを作成する方法、このデータを保持する表を作成する方法、表に DB2 監査データを入れる方法、および表から DB2 監査データを選択する方法について説明します。

DB2 表での DB2 監査データの操作

DB2 監査機能を使用してデータベース・アクティビティーの監査履歴を保守するとき、デフォルトでは、監査レコードがログ・ファイルに格納されます。必要に応じて、監査レコードをログ・ファイルからテキスト・ファイルに書き込んだり、区切り文字付き ASCII ファイルに書き込んで ASCII ファイルの内容を DB2 表にロードすることができます。監査データを DB2 表に取り込めば、表からデータを選択することにより、 DB2 インスタンスのアクティビティーをより良く観察できます。

手順:

監査データを DB2 表で操作するには、以下のようにします。

1. DB2 監査データを保持する表の作成
2. DB2 監査データ・ファイルの作成
3. ロード・ユーティリティーを使用して、表にデータを入れる
4. 表データの選択

関連概念:

- 283 ページの『監査機能の動作』
- 315 ページの『監査機能のヒントと技法』

関連タスク:

- 292 ページの『DB2 監査データ・ファイルの作成』
- 289 ページの『DB2 監査データを保持する表の作成』
- 293 ページの『DB2 監査データの表へのロード』
- 296 ページの『表での DB2 監査データの選択』

関連資料:

- 285 ページの『監査機能の使用方法』

DB2 監査データを保持する表の作成

監査データを表で操作する前に、データを保持するための表を作成する必要があります。表のデータを無許可ユーザーから保護するために、これらの表を 1 つの別個のスキーマで作成することを考慮してください。

前提条件:

- スキーマの作成に必要な権限および特権については、 CREATE SCHEMA ステートメントの説明を参照してください。
- 表の作成に必要な権限および特権については、 CREATE TABLE ステートメントの説明を参照してください。
- 表を保持するために、どの表スペースを使用するかを決定します。 (このトピックでは、表スペースの作成方法については説明しません。)

手順:

以下のいくつかの例は、すべての ASCII ファイルに含まれるすべてのレコードを保持する表を作成する方法を示しています。必要に応じて、これらの表を格納するための 1 つの別個のスキーマを作成することもできます。

ファイルに含まれるすべてのデータを必ずしも使用する必要がない場合には、表の定義から列を省略するか、必要に応じて、表の作成を回避することができます。表の定義から列を省略した場合、これらの表にデータをロードするためのコマンドを変更する必要があります。

1. **db2** コマンドを発行して、 DB2 コマンド・ウィンドウをオープンします。
2. オプション。表を保持するためのスキーマを作成します。以下のコマンドを発行してください。この例では、スキーマの名前は AUDIT です。

```
CREATE SCHEMA AUDIT
```

3. オプション。 AUDIT スキーマを作成した場合には、表を作成する前に、そのスキーマに切り替えます。以下のコマンドを発行してください。

```
SET CURRENT SCHEMA = 'AUDIT'
```

4. audit.del ファイルのレコードを格納する表を作成するには、以下の SQL ステートメントを発行します。

```
CREATE TABLE AUDIT (TIMESTAMP CHAR(26),
                    CATEGORY CHAR(8),
                    EVENT VARCHAR(32),
                    CORRELATOR INTEGER,
                    STATUS INTEGER,
                    USERID VARCHAR(1024),
                    AUTHID VARCHAR(128))
```

5. checking.del ファイルのレコードを格納する表を作成するには、以下の SQL ステートメントを発行します。

```
CREATE TABLE CHECKING (TIMESTAMP CHAR(26),
                       CATEGORY CHAR(8),
                       EVENT VARCHAR(32),
                       CORRELATOR INTEGER,
                       STATUS INTEGER,
                       DATABASE CHAR(8),
                       USERID VARCHAR(1024),
                       AUTHID VARCHAR(128),
                       NODENUM SMALLINT,
                       COORDNUM SMALLINT,
                       APPID VARCHAR(255),
                       APPNAME VARCHAR(1024),
                       PKGSCHEMA VARCHAR(128),
                       PKGNAME VARCHAR(128),
                       PKGSECNUM SMALLINT,
                       OBJSCHEMA VARCHAR(128),
                       OBJNAME VARCHAR(128),
                       OBJTYPE VARCHAR(32),
                       ACCESSAPP CHAR(18),
                       ACCESSATT CHAR(18),
                       PKGVER VARCHAR(64))
```

6. objmaint.del ファイルのレコードを格納する表を作成するには、以下の SQL ステートメントを発行します。

```
CREATE TABLE OBJMAINT (TIMESTAMP CHAR(26),
                       CATEGORY CHAR(8),
                       EVENT VARCHAR(32),
                       CORRELATOR INTEGER,
                       STATUS INTEGER,
                       DATABASE CHAR(8),
                       USERID VARCHAR(1024),
                       AUTHID VARCHAR(128),
                       NODENUM SMALLINT,
                       COORDNUM SMALLINT,
                       APPID VARCHAR(255),
                       APPNAME VARCHAR(1024),
                       PKGSCHEMA VARCHAR(128),
                       PKGNAME VARCHAR(128),
                       PKGSECNUM SMALLINT,
                       OBJSCHEMA VARCHAR(128),
                       OBJNAME VARCHAR(128),
                       OBJTYPE VARCHAR(32),
                       PACKVER VARCHAR(64))
```

7. secmaint.del ファイルのレコードを格納する表を作成するには、以下の SQL ステートメントを発行します。

```
CREATE TABLE SECMAINT (TIMESTAMP CHAR(26),
                       CATEGORY CHAR(8),
                       EVENT VARCHAR(32),
                       CORRELATOR INTEGER,
                       STATUS INTEGER,
                       DATABASE CHAR(8),
                       USERID VARCHAR(1024),
                       AUTHID VARCHAR(128),
                       NODENUM SMALLINT,
                       COORDNUM SMALLINT,
                       APPID VARCHAR(255),
                       APPNAME VARCHAR(1024),
                       PKGSCHEMA VARCHAR(128),
                       PKGNAME VARCHAR(128),
                       PKGSECNUM SMALLINT,
                       OBJSCHEMA VARCHAR(128),
                       OBJNAME VARCHAR(128),
                       OBJTYPE VARCHAR(32),
                       GRANTOR VARCHAR(128),
                       GRANTEE VARCHAR(128),
                       GRANTEETYPE VARCHAR(32),
                       PRIVAUTH CHAR(18),
                       PKGVER VARCHAR(64))
```

8. sysadmin.del ファイルのレコードを格納する表を作成するには、以下の SQL ステートメントを発行します。

```
CREATE TABLE SYSADMIN (TIMESTAMP CHAR(26),
                       CATEGORY CHAR(8),
                       EVENT VARCHAR(32),
                       CORRELATOR INTEGER,
                       STATUS INTEGER,
                       DATABASE CHAR(8),
                       USERID VARCHAR(1024),
                       AUTHID VARCHAR(128),
                       NODENUM SMALLINT,
                       COORDNUM SMALLINT,
                       APPID VARCHAR(255),
                       APPNAME VARCHAR(1024),
                       PKGSCHEMA VARCHAR(128),
                       PKGNAME VARCHAR(128),
                       PKGSECNUM SMALLINT,
                       PKGVER VARCHAR(64))
```

9. validate.del ファイルのレコードを格納する表を作成するには、以下の SQL ステートメントを発行します。

```
CREATE TABLE VALIDATE (TIMESTAMP CHAR(26),
                       CATEGORY CHAR(8),
                       EVENT VARCHAR(32),
                       CORRELATOR INTEGER,
                       STATUS INTEGER,
                       DATABASE CHAR(8),
                       USERID VARCHAR(1024),
                       AUTHID VARCHAR(128),
                       EXECID VARCHAR(1024),
                       NODENUM SMALLINT,
                       COORDNUM SMALLINT,
                       APPID VARCHAR(255),
                       APPNAME VARCHAR(1024),
                       AUTHTYPE VARCHAR(32),
                       PKGSCHEMA VARCHAR(128),
                       PKGNAME VARCHAR(128),
                       PKGSECNUM SMALLINT,
                       PKGVER VARCHAR(64)
                       PLUGINNAME VARCHAR(32))
```

10. context.del ファイルのレコードを格納する表を作成するには、以下の SQL ステートメントを発行します。

```
CREATE TABLE CONTEXT (TIMESTAMP CHAR(26),
                      CATEGORY CHAR(8),
                      EVENT VARCHAR(32),
                      CORRELATOR INTEGER,
                      DATABASE CHAR(8),
                      USERID VARCHAR(1024),
                      AUTHID VARCHAR(128),
                      NODENUM SMALLINT,
                      COORDNUM SMALLINT,
                      APPID VARCHAR(255),
                      APPNAME VARCHAR(1024),
                      PKGSCHEMA VARCHAR(128),
                      PKGNAME VARCHAR(128),
                      PKGSECNUM SMALLING,
                      STMTTEXT CLOB(2M),
                      PKGVER VARCHAR(64))
```

11. これらの表を作成した後、表定義をディスクに書き込むために、 COMMIT ステートメントを発行してください。
12. 表を作成した後、 監査レコードを db2audit.log ファイルから区切り文字付き ASCII ファイルに抽出します。

関連タスク:

- 96 ページの『スキーマの設定』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『CREATE SCHEMA ステートメント』
- 「*SQL リファレンス 第 2 巻*」の『CREATE TABLE ステートメント』

DB2 監査データ・ファイルの作成

デフォルトでは、監査データは DB2 監査機能によって db2audit.log ファイルに書き込まれます。このファイル内のレコードを表にロードすることはできません。監査レコードを区切り文字付き ASCII に抽出すれば、それを使用して表にデータを入れることができます。

前提条件:

db2audit コマンドを使用するには、 SYSADM 権限が必要です。

手順:

監査機能レコードを区切り文字付き ASCII ファイルに書き込むには、以下のようにします。

1. 監査機能の使用方法に関するトピックを見て、どの DB2 アクティビティーを監査するかを決定します。監査機能の構成を適切にセットアップした後、以下のコマンドを発行して監査を開始します。

```
db2audit start
```

2. すべての監査レコードをメモリーから db2audit.log ファイルにフラッシュするために、以下のコマンドを発行します。

```
db2audit flush
```

3. 監査レコードを db2audit.log から区切り文字付き ASCII ファイルに移動するために、以下のコマンドを発行します。

   ```
   db2audit extract delasc
   ```

   sqllib の security サブディレクトリーに、以下のファイルが作成されます。いずれかのタイプのイベントを監査しない場合、そのイベントのファイルは作成されますが、ファイルの中身は空です。
   - audit.del
   - checking.del
   - objmaint.del
   - secmaint.del
   - sysadmin.del
   - validate.del
   - context.del
4. すでに抽出した db2audit.log ファイルから監査レコードを削除するには、以下のコマンドを発行します。

   ```
   db2audit prune date YYYYMMDDHH
   ```

   ここで、*YYYYMMDDHH* は現在の年月日と時刻です。この値は、次のステップで表に監査データを入れるときに必要とされるため、この値を書き留めてください。

   監査機能は、新しい監査レコードを db2audit.log ファイルに書き込み続けます。そのようにレコードのタイム・スタンプは、 *YYYYMMDDHH* より大きくなります。すでに抽出済みのレコードを db2audit.log ファイルから整理すれば、同じレコードを次回に抽出するのを防ぐことができます。次に監査レコードを抽出するときには、 *YYYYMMDDHH* より後に書き込まれたすべての監査レコードが .del ファイルに書き込まれるようになります。
5. 監査データ・ファイルを作成した後は、 ロード・ユーティリティーを使用して、表に監査データを入れるステップに進みます。

関連資料:

- 「コマンド・リファレンス」の『db2audit - 監査機能管理者ツール・コマンド』
- 285 ページの『監査機能の使用方法』
- 298 ページの『AUDIT イベントの監査レコード設計』
- 299 ページの『CHECKING イベントの監査レコード設計』
- 305 ページの『OBJMAINT イベントの監査レコード設計』
- 306 ページの『SECMAINT イベントの監査レコード設計』
- 310 ページの『SYSADMIN イベントの監査レコード設計』
- 313 ページの『VALIDATE イベントの監査レコード設計』
- 314 ページの『CONTEXT イベントの監査レコード設計』

DB2 監査データの表へのロード

監査データを保持するための表を作成した後、 ASCII ファイル内のデータをロードします。

前提条件:

詳しくは、 ロード・ユーティリティーを使用するのに必要な特権、権限、および許可のトピックを参照してください。

手順:

ロード・ユーティリティーを使用して、データを表にロードします。それぞれの表ごとに、別個のロード・コマンドを発行してください。表定義から 1 つまたは複数の列を省略した場合には、データを正常にロードするために、LOAD コマンドの内容を変更する必要があります。さらに、監査データの抽出時にデフォルト (0xff) 以外の区切り文字を指定した場合にもまた、使用する LOAD コマンドの内容を変更する必要があります (詳しくは、『ロードのファイル・タイプ修飾子』のトピックを参照)。

1. **db2** コマンドを発行して、 DB2 コマンド・ウィンドウをオープンします。
2. AUDIT 表をロードするために、以下のコマンドを発行します。

   ```
   LOAD FROM audit.del OF del MODIFIED BY CHARDEL0xff INSERT INTO schema.AUDIT
   ```

   注: ファイル名を指定するときには、完全修飾パスを使用してください。たとえば、Windows コンピューターの C: ドライブに DB2 UDB がインストールされている場合、 audit.del ファイルの完全修飾ファイル名として、C:¥Program Files¥IBM¥SQLLIB¥*instance*¥security¥audit.del と指定します。

   AUDIT 表をロードした後、次回のロード時に重複する行を表にロードしないようにするために、以下の DELETE ステートメントを発行してください。監査レコードを db2audit.log ファイルから抽出したとき、ファイル内のすべてのレコードが .del ファイルに書き込まれました。 .del ファイルには、監査ログが整理された時点以降に書き込まれたレコードが含まれると予想されます (**db2audit prune** コマンドは特定の時刻までのレコードを整理します)。次回に監査レコードを抽出するとき、新しい .del ファイルには、すでに抽出されたものの **db2audit prune** コマンドによって整理されていないレコードが含まれます (これらのレコードは、整理操作で指定された時刻以降に書き込まれたためです)。 db2audit.log ファイルが整理された時刻と同じ時点まで表から行を削除すれば、重複する行が表に含まれず、監査レコードが失われることもありません。

   ```
   DELETE FROM schema.AUDIT WHERE TIMESTAMP > TIMESTAMP('YYYYMMDDHH0000')
   ```

   ここで、*YYYYMMDDHH* は、 db2audit.log ファイルを整理したときに指定した値です。 DB2 監査機能は db2audit.log ファイルの整理後もこのファイルに監査レコードを書き込み続けるため、 db2audit.log ファイルの整理後に書き込まれた監査レコードが表から削除されないようにするために、分および秒に 0000 を指定する必要があります。
3. CHECKING 表をロードするために、以下のコマンドを発行します。

   ```
   LOAD FROM checking.del OF del MODIFIED BY CHARDEL0xff INSERT INTO
     schema.CHECKING
   ```

   CHECKING 表をロードした後、次回のロード時に重複する行を表にロードしないようにするために、以下の SQL ステートメントを発行してください。

```
DELETE FROM schema.CHECKING WHERE TIMESTAMP > TIMESTAMP('YYYYMMDDHH0000')
```

ここで、*YYYYMMDDHH* は、ログ・ファイルを整理したときに指定した値です。

4. OBJMAINT 表をロードするために、以下のコマンドを発行します。

```
LOAD FROM objmaint.del OF del MODIFIED BY CHARDEL0xff INSERT INTO
  schema.OBJMAINT
```

OBJMAINT 表をロードした後、次回のロード時に重複する行を表にロードしないようにするために、以下の SQL ステートメントを発行してください。

```
DELETE FROM schema.OBJMAINT WHERE TIMESTAMP > TIMESTAMP('YYYYMMDDHH0000')
```

ここで、*YYYYMMDDHH* は、ログ・ファイルを整理したときに指定した値です。

5. SECMAINT 表をロードするために、以下のコマンドを発行します。

```
LOAD FROM secmaint.del OF del MODIFIED BY CHARDEL0xff INSERT INTO
  schema.SECMAINT
```

SECMAINT 表をロードした後、次回のロード時に重複する行を表にロードしないようにするために、以下の SQL ステートメントを発行してください。

```
DELETE FROM schema.SECMAINT WHERE TIMESTAMP > TIMESTAMP('YYYYMMDDHH0000')
```

ここで、*YYYYMMDDHH* は、ログ・ファイルを整理したときに指定した値です。

6. SYSADMIN 表をロードするために、以下のコマンドを発行します。

```
LOAD FROM sysadmin.del OF del MODIFIED BY CHARDEL0xff INSERT INTO
  schema.SYSADMIN
```

SYSADMIN 表をロードした後、次回のロード時に重複する行を表にロードしないようにするために、以下の SQL ステートメントを発行してください。

```
DELETE FROM schema.SYSADMIN WHERE TIMESTAMP > TIMESTAMP('YYYYMMDDHH0000')
```

ここで、*YYYYMMDDHH* は、ログ・ファイルを整理したときに指定した値です。

7. VALIDATE 表をロードするために、以下のコマンドを発行します。

```
LOAD FROM validate.del OF del MODIFIED BY CHARDEL0xff INSERT INTO
  schema.VALIDATE
```

VALIDATE 表をロードした後、次回のロード時に重複する行を表にロードしないようにするために、以下の SQL ステートメントを発行してください。

```
DELETE FROM schema.VALIDATE WHERE TIMESTAMP > TIMESTAMP('YYYYMMDDHH0000')
```

ここで、*YYYYMMDDHH* は、ログ・ファイルを整理したときに指定した値です。

8. CONTEXT 表をロードするために、以下のコマンドを発行します。

```
LOAD FROM context.del OF del MODIFIED BY CHARDEL0xff INSERT INTO
  schema.CONTEXT
```

CONTEXT 表をロードした後、次回のロード時に重複する行を表にロードしないようにするために、以下の SQL ステートメントを発行してください。

```
DELETE FROM schema.CONTEXT WHERE TIMESTAMP > TIMESTAMP('YYYYMMDDHH0000')
```

ここで、*YYYYMMDDHH* は、ログ・ファイルを整理したときに指定した値です。

9. これらの表にデータをロードした後、 sqllib ディレクトリーの security サブディレクトリーから .del ファイルを削除します。
10. 監査データを表にロードした後、 これらの表からデータを選択します。

表にあらかじめデータが入っていて、再びデータを入れたい場合には、既存の表データに新しい表データを追加するために、 INSERT オプションを使用してください。以前の **db2audit extract** 操作によるレコードを表から除去したい場合には、REPLACE オプションを使って表を再びロードします。どちらの場合も、監査レコードを .del ファイルに抽出する前に、必ずレコードを db2audit.log ファイルにフラッシュしてください。さらに、同じレコードを表に 2 度以上ロードしないようにするために、レコードの抽出後、必ず db2audit.log ファイルを整理してください。

関連タスク:

- 288 ページの『DB2 表での DB2 監査データの操作』

表での DB2 監査データの選択

監査データを正常に表にロードした後、これらの表のデータを選択して、詳しく分析することができます。

前提条件:

表でデータを選択するために必要な権限および特権については、 SELECT ステートメントに関するトピックを参照してください。

手順:

表のすべての行を選択するには、以下のようにします。

1. **db2** コマンドを発行して、 DB2 コマンド・ウィンドウをオープンします。
2. 監査データを選択するそれぞれの表に対して、以下の形式の SQL ステートメントを発行します。

```
SELECT * FROM schema.table
```

たとえば、AUDIT スキーマ内の CHECKING 表に含まれるすべてのデータを選択するには、以下のステートメントを使用します。

```
SELECT * FROM AUDIT.CHECKING
```

選択内容は、データに対してどのような分析をしたいかに応じて異なります。たとえば、特定の許可 ID (authid) がどんなアクティビティーを実行したかを判別するには、この許可 ID に基づいて以下のようにレコードを選択できます。

```
SELECT * FROM AUDIT.CHECKING WHERE AUTHID = authorization ID
```

ここで、*authorization ID* (許可 ID) はデータ分析対象のユーザー ID です。

監査データに含まれるさまざまな値についての説明は、表におけるそれぞれの監査レコード設計のトピック、および表での戻り値のリストを参照してください。

関連資料:

- 「*SQL リファレンス 第 1 巻*」の『副選択』
- 「*SQL リファレンス 第 2 巻*」の『SELECT ステートメント』
- 298 ページの『AUDIT イベントの監査レコード設計』
- 299 ページの『CHECKING イベントの監査レコード設計』
- 302 ページの『有効な CHECKING アクセス承認理由のリスト』
- 303 ページの『有効な CHECKING アクセス試行タイプのリスト』
- 305 ページの『OBJMAINT イベントの監査レコード設計』
- 306 ページの『SECMAINT イベントの監査レコード設計』
- 307 ページの『有効な SECMAINT 特権または権限のリスト』
- 310 ページの『SYSADMIN イベントの監査レコード設計』
- 311 ページの『有効な SYSADMIN 監査イベントのリスト』
- 313 ページの『VALIDATE イベントの監査レコード設計』
- 314 ページの『CONTEXT イベントの監査レコード設計』
- 314 ページの『有効な CONTEXT 監査イベントのリスト』

監査機能のメッセージ

SQL1322N 監査ログ・ファイルへの書き込み時にエラーが発生しました。

説明: 監査イベントを監査ログ・ファイルに記録するために呼び出すとき、 DB2 Universal Database™ (DB2 UDB) 監査機能はエラーを検出しました。監査ログがあるファイル・システムにはスペースがありません。

ユーザーの処置: システム管理者は、このファイル・システムのスペースを空けるか、または監査ログのサイズを削減するために余分なものを取り除くようにすべきです。

より大きなスペースが使用可能なときに、 db2audit を使用してメモリー内のすべてのデータをフラッシュアウトし、監査機能を作動可能状態にリセットしてください。削除済みレコードはリカバリーできないため、適切な抽出を実行するか、ログの整理の前にログのコピーを必ず実行してください。

sqlcode: -1322

sqlstate: 50830

SQL1323N 監査構成ファイルにアクセスしているときエラーが発生しました。

説明: 監査構成ファイル (db2audit.cfg) を開くことができないか、または無効でした。このエラーの考えられる理由は、db2audit.cfg ファイルが存在しないか、または損傷したかのいずれかです。

ユーザーの処置: 下記の処置のいずれかを行ってください。

- 保管されたバージョンのファイルからリストアする。
- 以下を発行して、監査機能の構成ファイルをリセットする。

```
db2audit reset
```

sqlcode: -1323

sqlstate: 57019

関連概念:

- 281 ページの『DB2 Universal Database (DB2 UDB) 監査機能の紹介』

監査機能のレコード設計の紹介

監査レコードが DELASC 抽出オプションを使用して監査ログから抽出されるとき、それぞれのレコードは下記の表で示されるいずれかのフォーマットになります。それぞれの表のはじめに、サンプル・レコードの内容を示します。レコードの各項目の記述は、関連する表において一度に 1 つの行で示されます。その項目が重要である場合、項目の名前は強調表示 (**太字**) されています。これらの項目には、ユーザーにとって非常に重要な情報が含まれています。

注:

1. サンプル・レコードのすべてのフィールドが、必ずしも値を持っているとは限りません。
2. 中には、″Access Attempted″ のようにビットマップとして区切り文字付き ASCII 形式で保管されるフィールドもあります。ただし、現在のフラットなレポート・ファイルにおいて、それらのフィールドはビットマップ値を表す一連のストリングとして表示されます。
3. ″Package Version″ という新規のフィールドが、CHECKING、OBJMAINT、SECMAINT、SYSADMIN、VALIDATE、および CONTEXT イベントに対して追加されました。

関連資料:

- 298 ページの『AUDIT イベントの監査レコード設計』
- 299 ページの『CHECKING イベントの監査レコード設計』
- 305 ページの『OBJMAINT イベントの監査レコード設計』
- 306 ページの『SECMAINT イベントの監査レコード設計』
- 310 ページの『SYSADMIN イベントの監査レコード設計』
- 313 ページの『VALIDATE イベントの監査レコード設計』
- 314 ページの『CONTEXT イベントの監査レコード設計』

監査機能のレコード設計に関する詳細

このセクションでは、さまざまな監査機能のレコード設計を示します。

AUDIT イベントの監査レコード設計

表 7. AUDIT イベントの監査レコード設計

timestamp=1998-06-24-11.54.05.151232;**category=AUDIT;audit event=START;**
event correlator=0;event status=0;
userid=boss;authid=BOSS;		

NAME	**フォーマット**	**記述**
Timestamp	CHAR(26)	監査イベントの日付と時刻。
Category	CHAR(8)	監査イベントの区分。可能な値は以下のとおりです。 AUDIT

表 7. AUDIT イベントの監査レコード設計 (続き)

```
timestamp=1998-06-24-11.54.05.151232;category=AUDIT;audit event=START;
  event correlator=0;event status=0;
  userid=boss;authid=BOSS;
```

NAME	フォーマット	記述
Audit Event	VARCHAR(32)	特定の監査イベント。 有効な値には、 CONFIGURE、 DB2AUD、 EXTRACT、 FLUSH、 PRUNE、 START、 STOP、および UPDATE_ADMIN_CFG が含まれます。
Event Correlator	INTEGER	監査対象の操作の相関 ID。単一イベントにどの監査レコードが関連しているかを識別するために使用できます。
Event Status	INTEGER	監査イベントの状況で、次のような 1 つの SQLCODE で表されます。 成功イベント >= 0 失敗イベント < 0
User ID	VARCHAR(1024)	監査イベントの時刻でのユーザー ID。
Authorization ID	VARCHAR(128)	監査イベントの時刻での許可 ID。

関連概念:

- 298 ページの『監査機能のレコード設計の紹介』

CHECKING イベントの監査レコード設計

表 8. CHECKING イベントの監査レコード設計

```
timestamp=1998-06-24-08.42.11.622984;category=CHECKING;audit event=CHECKING_OBJECT;
  event correlator=2;event status=0;
  database=FOO;userid=boss;authid=BOSS;
  application id=*LOCAL.newton.980624124210;application name=testapp;
  package schema=NULLID;package name=SYSSH200;
  package section=0;object schema=GSTAGER;object name=NONE;object type=REOPT_VALUES;
  access approval reason=DBADM;access attempted=STORE;
```

名前	フォーマット	記述
Timestamp	CHAR(26)	監査イベントの日付と時刻。
Category	CHAR(8)	監査イベントの区分。可能な値は以下のとおりです。 CHECKING
Audit Event	VARCHAR(32)	特定の監査イベント。 有効な値は、CHECKING_OBJECT および CHECKING_FUNCTION です。
Event Correlator	INTEGER	監査対象の操作の相関 ID。単一イベントにどの監査レコードが関連しているかを識別するために使用できます。
Event Status	INTEGER	監査イベントの状況で、次のような 1 つの SQLCODE で表されます。 成功イベント >= 0 失敗イベント < 0

表 8. *CHECKING* イベントの監査レコード設計 (続き)

```
timestamp=1998-06-24-08.42.11.622984;category=CHECKING;audit event=CHECKING_OBJECT;
  event correlator=2;event status=0;
  database=FOO;userid=boss;authid=BOSS;
  application id=*LOCAL.newton.980624124210;application name=testapp;
  package schema=NULLID;package name=SYSSH200;
  package section=0;object schema=GSTAGER;object name=NONE;object type=REOPT_VALUES;
  access approval reason=DBADM;access attempted=STORE;
```

名前	フォーマット	記述
Database Name	CHAR(8)	どのイベントが生成されたかを示すデータベース名。インスタンス・レベルの監査イベントであった場合にはブランクとなります。
User ID	VARCHAR(1024)	監査イベントの時刻でのユーザー ID。
Authorization ID	VARCHAR(128)	監査イベントの時刻での許可 ID。
Origin Node Number	SMALLINT	監査イベントが発生したデータベース・パーティション番号。
Coordinator Node Number	SMALLINT	コーディネーター・データベース・パーティションのデータベース・パーティション番号。
Application ID	VARCHAR (255)	監査イベントが発生した時刻で使用していたアプリケーション ID。
Application Name	VARCHAR (1024)	監査イベントが発生した時刻で使用していたアプリケーション名。
Package Schema	VARCHAR (128)	監査イベントの時刻で使用していたパッケージのスキーマ。
Package Name	VARCHAR (128)	監査イベントが発生した時刻で使用していたパッケージ名。
Package Section Number	SMALLINT	監査イベントが発生した時刻で使用されていたパッケージのセクション番号。
Object Schema	VARCHAR (128)	監査イベントの生成対象となったオブジェクトのスキーマ。
Object Name	VARCHAR (128)	監査イベントの生成対象となったオブジェクトの名前。
Object Type	VARCHAR (32)	監査イベントの生成対象となったオブジェクトのタイプ。有効な値は、『監査レコード・オブジェクト・タイプ』というトピックに示されています。
Access Approval Reason	CHAR(18)	アクセスがこの監査イベントで承認された理由を示します。有効な値は、『有効な CHECKING アクセス承認理由のリスト』というトピックに示されています。
Access Attempted	CHAR(18)	試みられたアクセスのタイプを示します。有効な値は、『有効な CHECKING アクセス試行タイプのリスト』というトピックに示されています。
Package Version	VARCHAR (64)	監査イベントが発生した時点で使用されていたパッケージのバージョン。

関連概念:

- 298 ページの『監査機能のレコード設計の紹介』

関連資料:

- 302 ページの『有効な CHECKING アクセス承認理由のリスト』
- 303 ページの『有効な CHECKING アクセス試行タイプのリスト』
- 301 ページの『監査レコード・オブジェクト・タイプ』

監査レコード・オブジェクト・タイプ

表 9. 監査イベントに基づく監査レコード・オブジェクト・タイプ

オブジェクト・タイプ	CHECKING イベント	OBJMAINT イベント	SECMAINT イベント
NONE	○	○	○
TABLE	○	○	○
VIEW	○	○	○
ALIAS	○	○	
FUNCTION	○	○	○
INDEX	○	○	○
INDEX EXTENSION		○	
PACKAGE	○	○	○
PACKAGE CACHE	○		
DATA_TYPE		○	
NODEGROUP	○	○	
SCHEMA	○	○	○
STORED_PROCEDURE	○	○	○
METHOD_BODY	○	○	○
BUFFERPOOL	○	○	
SEQUENCE	○	○	
TABLESPACE	○	○	○
EVENT_MONITOR	○	○	
TRIGGER		○	
DATABASE	○		○
INSTANCE	○		
FOREIGN_KEY		○	
PRIMARY_KEY		○	
UNIQUE_CONSTRAINT		○	
CHECK_CONSTRAINT		○	
WRAPPER	○	○	
SERVER	○	○	○
NICKNAME	○	○	○
USER MAPPING	○	○	
SERVER OPTION	○	○	
TYPE&TRANSFORM	○	○	
TYPE MAPPING	○	○	
FUNCTION MAPPING	○	○	
SUMMARY TABLES	○	○	○
JAR_FILE		○	
ALL	○		
REOPT_VALUES	○		

関連資料:

- 299 ページの『CHECKING イベントの監査レコード設計』
- 305 ページの『OBJMAINT イベントの監査レコード設計』
- 306 ページの『SECMAINT イベントの監査レコード設計』

有効な CHECKING アクセス承認理由のリスト

下記は、有効な CHECKING アクセス承認理由のリストです。

0x0000000000000001 ACCESS DENIED
アクセスは承認されません。その上、拒否されました。

0x0000000000000002 SYSADM
アクセスは承認されます。アプリケーション/ユーザーは SYSADM 権限を持ちます。

0x0000000000000004 SYSCTRL
アクセスは承認されます。アプリケーション/ユーザーは SYSCTRL 権限を持ちます。

0x0000000000000008 SYSMAINT
アクセスは承認されます。アプリケーション/ユーザーは SYSMAINT 権限を持ちます。

0x0000000000000010 DBADM
アクセスは承認されます。アプリケーション/ユーザーは DBADM 権限を持ちます。

0x0000000000000020 DATABASE PRIVILEGE
アクセスは承認されます。アプリケーション/ユーザーはこのデータベースに関して明示的な権限を持ちます。

0x0000000000000040 OBJECT PRIVILEGE
アクセスは承認されます。アプリケーション/ユーザーはオブジェクトまたは関数に関して明示的な権限を持ちます。

0x0000000000000080 DEFINER
アクセスは承認されます。アプリケーション/ユーザーは、オブジェクトまたは関数の定義をするものとなります。

0x0000000000000100 OWNER
アクセスは承認されます。アプリケーション/ユーザーは、オブジェクトまたは関数の所有者となります。

0x0000000000000200 CONTROL
アクセスは承認されます。アプリケーション/ユーザーは、オブジェクトまたは関数に関する CONTROL 権限を持ちます。

0x0000000000000400 BIND
アクセスは承認されます。アプリケーション/ユーザーは、パッケージに関するバインド権限を持ちます。

0x0000000000000800 SYSQUIESCE
アクセスは承認されます。インスタンスまたはデータベースが静止モードにある場合は、アプリケーション/ユーザーは接続またはアタッチを行うことができます。

0x0000000000001000 SYSMON
アクセスは承認されます。アプリケーション/ユーザーは SYSMON 権限を持ちます。

関連資料:

- 299 ページの『CHECKING イベントの監査レコード設計』
- 303 ページの『有効な CHECKING アクセス試行タイプのリスト』

有効な CHECKING アクセス試行タイプのリスト

以下は、有効な CHECKING アクセス試行タイプのリストです。

0x0000000000000002 ALTER
オブジェクトを変更しようとします。

0x0000000000000004 DELETE
オブジェクトを削除しようとします。

0x0000000000000008 INDEX
索引を使用しようとします。

0x0000000000000010 INSERT
オブジェクトの中に挿入しようとします。

0x0000000000000020 SELECT
表またはビューを照会しようとします。

0x0000000000000040 UPDATE
オブジェクトのデータを更新しようとします。

0x0000000000000080 REFERENCE
オブジェクト間の参照制約を確立しようとします。

0x0000000000000100 CREATE
オブジェクトを作成しようとします。

0x0000000000000200 DROP
オブジェクトをドロップしようとします。

0x0000000000000400 CREATEIN
別のスキーマ内にオブジェクトを作成しようとします。

0x0000000000000800 DROPIN
別のスキーマ内に見いだされるオブジェクトをドロップしようとします。

0x0000000000001000 ALTERIN
別のスキーマ内に見いだされるオブジェクトを変更しようとします。

0x0000000000002000 EXECUTE
アプリケーションを実行、またはルーチンを呼び出そうとします。ルーチンからのソース関数を作成する (関数のみに適用されます) か、何らかの DDL ステートメントのルーチンを参照します。

0x0000000000004000 BIND
アプリケーションをバインドまたは準備しようとします。

0x0000000000008000 SET EVENT MONITOR
イベント・モニターのスイッチをセットしようとします。

0x0000000000010000 SET CONSTRAINTS
オブジェクトに関する制約をセットしようとします。

0x0000000000020000 COMMENT ON
オブジェクトに関する注釈を作成しようとします。

0x0000000000040000 GRANT
別のユーザー ID にオブジェクトに関する特権を付与しようとします。

0x0000000000080000 REVOKE
オブジェクトに関する特権をユーザー ID から取り消そうとします。

0x0000000000100000 LOCK
オブジェクトをロックしようとします。

0x0000000000200000 RENAME
オブジェクトを名前変更しようとします。

0x0000000000400000 CONNECT
オブジェクトに接続しようとします。

0x0000000000800000 Member of SYS Group
SYS グループのメンバーをアクセスまたは使用しようとします。

0x0000000001000000 Access All
保持されているオブジェクトに対して必要なすべての特権を使用して、ステートメントを実行しようとします (DBADM/SYSADM でのみ使用されます)。

0x0000000002000000 Drop All
複数のオブジェクトをドロップしようとします。

0x0000000004000000 LOAD
表スペースに表をロードしようとします。

0x0000000008000000 USE
表スペースに表を作成しようとします。

0x0000000010000000 SET SESSION_USER
SET SESSION_USER ステートメントを実行しようとします。

0x0000000020000000 FLUSH
FLUSH ステートメントを実行しようとします。

0x0000000040000000 STORE
EXPLAIN_PREDICATE 表内の再び最適化されたステートメントの値を表示しようとします。

関連資料:

- 299 ページの『CHECKING イベントの監査レコード設計』
- 302 ページの『有効な CHECKING アクセス承認理由のリスト』

OBJMAINT イベントの監査レコード設計

表 10. OBJMAINT イベントの監査レコード設計

timestamp=1998-06-24-08.42.41.957524;**category=OBJMAINT;audit event=CREATE_OBJECT;**
event correlator=3;event status=0;
database=FOO;userid=boss;authid=BOSS;
application id=*LOCAL.newton.980624124210;application name=testapp;
package schema=NULLID;package name=SQLC28A1;
package section=0;object schema=BOSS;object name=AUDIT;object type=TABLE;		

名前	**フォーマット**	**記述**
Timestamp	CHAR(26)	監査イベントの日付と時刻。
Category	CHAR(8)	監査イベントの区分。可能な値は以下のとおりです。 OBJMAINT
Audit Event	VARCHAR(32)	特定の監査イベント。 有効な値は、CREATE_OBJECT、RENAME_OBJECT、および DROP_OBJECT です。
Event Correlator	INTEGER	監査対象の操作の相関 ID。単一イベントにどの監査レコードが関連しているかを識別するために使用できます。
Event Status	INTEGER	監査イベントの状況、次のような 1 つの SQLCODE で表されます。 成功イベント >= 0 失敗イベント < 0
Database Name	CHAR(8)	どのイベントが生成されたかを示すデータベース名。インスタンス・レベルの監査イベントであった場合にはブランクとなります。
User ID	VARCHAR(1024)	監査イベントの時刻でのユーザー ID。
Authorization ID	VARCHAR(128)	監査イベントの時刻での許可 ID。
Origin Node Number	SMALLINT	監査イベントが発生したデータベース・パーティション番号。
Coordinator Node Number	SMALLINT	コーディネーター・データベース・パーティションのデータベース・パーティション番号。
Application ID	VARCHAR (255)	監査イベントが発生した時刻で使用していたアプリケーション ID。
Application Name	VARCHAR (1024)	監査イベントが発生した時刻で使用していたアプリケーション名。
Package Schema	VARCHAR (128)	監査イベントの時刻で使用していたパッケージのスキーマ。
Package Name	VARCHAR (128)	監査イベントが発生した時刻で使用していたパッケージ名。
Package Section Number	SMALLINT	監査イベントが発生した時刻で使用されていたパッケージのセクション番号。
Object Schema	VARCHAR (128)	監査イベントの生成対象となったオブジェクトのスキーマ。
Object Name	VARCHAR (128)	監査イベントの生成対象となったオブジェクトの名前。
Object Type	VARCHAR (32)	監査イベントの生成対象となったオブジェクトのタイプ。有効な値は、『監査レコード・オブジェクト・タイプ』というトピックに示されています。
Package Version	VARCHAR (64)	監査イベントが発生した時刻で使用していたパッケージのバージョン。

関連概念:

- 281 ページの『DB2 Universal Database (DB2 UDB) 監査機能の紹介』

関連資料:

- 301 ページの『監査レコード・オブジェクト・タイプ』

SECMAINT イベントの監査レコード設計

表 11. SECMAINT イベントの監査レコード設計

```
timestamp=1998-06-24-11.57.45.188101;category=SECMAINT;audit event=GRANT;
  event correlator=4;event status=0;
  database=FOO;userid=boss;authid=BOSS;
  application id=*LOCAL.boss.980624155728;application name=db2bp;
  package schema=NULLID;package name=SQLC28A1;
  package section=0;object schema=BOSS;object name=T1;object type=TABLE;
  grantor=BOSS;grantee=WORKER;grantee type=USER;privilege=SELECT;
```

名前	フォーマット	記述
Timestamp	CHAR(26)	監査イベントの日付と時刻。
Category	CHAR(8)	監査イベントの区分。可能な値は以下のとおりです。 SECMAINT
Audit Event	VARCHAR(32)	特定の監査イベント。 有効な値を次に示します。GRANT、 REVOKE、 IMPLICIT_GRANT、 IMPLICIT_REVOKE、 SET_SESSION_USER、および UPDATE_DBM_CFG。
Event Correlator	INTEGER	監査対象の操作の相関 ID。単一イベントにどの監査レコードが関連しているかを識別するために使用できます。
Event Status	INTEGER	監査イベントの状況、次のような 1 つの SQLCODE で表されます。 成功イベント >= 0 失敗イベント < 0
Database Name	CHAR(8)	どのイベントが生成されたかを示すデータベース名。インスタンス・レベルの監査イベントであった場合にはブランクとなります。
User ID	VARCHAR(1024)	監査イベントの時刻でのユーザー ID。
Authorization ID	VARCHAR(128)	監査イベントの時刻での許可 ID。
Origin Node Number	SMALLINT	監査イベントが発生したデータベース・パーティション番号。
Coordinator Node Number	SMALLINT	コーディネーター・データベース・パーティションのデータベース・パーティション番号。
Application ID	VARCHAR (255)	監査イベントが発生した時刻で使用していたアプリケーション ID。
Application Name	VARCHAR (1024)	監査イベントが発生した時刻で使用していたアプリケーション名。
Package Schema	VARCHAR (128)	監査イベントの時刻で使用していたパッケージのスキーマ。
Package Name	VARCHAR (128)	監査イベントが発生した時刻で使用していたパッケージ名。
Package Section Number	SMALLINT	監査イベントが発生した時刻で使用されていたパッケージのセクション番号。
Object Schema	VARCHAR (128)	監査イベントの生成対象となったオブジェクトのスキーマ。
Object Name	VARCHAR (128)	監査イベントの生成対象となったオブジェクトの名前。
Object Type	VARCHAR (32)	監査イベントの生成対象となったオブジェクトのタイプ。有効な値は、『監査レコード・オブジェクト・タイプ』というトピックに示されています。

表 11. SECMAINT イベントの監査レコード設計 (続き)

```
timestamp=1998-06-24-11.57.45.188101;category=SECMAINT;audit event=GRANT;
  event correlator=4;event status=0;
  database=FOO;userid=boss;authid=BOSS;
  application id=*LOCAL.boss.980624155728;application name=db2bp;
  package schema=NULLID;package name=SQLC28A1;
  package section=0;object schema=BOSS;object name=T1;object type=TABLE;
  grantor=BOSS;grantee=WORKER;grantee type=USER;privilege=SELECT;
```

名前	フォーマット	記述
Grantor	VARCHAR (128)	認可者の ID。
Grantee	VARCHAR (128)	特権または権限が付与または取り消された被認可者の ID。
Grantee Type	VARCHAR (32)	付与または取り消された被認可者のタイプ。有効な値は、USER、GROUP、または BOTH です。
Privilege または Authority	CHAR(18)	付与または取り消された特権または権限のタイプを示します。有効な値は、『有効な SECMAINT 特権または権限のリスト』というトピックに示されています。
Package Version	VARCHAR (64)	監査イベントが発生した時刻で使用していたパッケージのバージョン。

関連概念:

- 298 ページの『監査機能のレコード設計の紹介』

関連資料:

- 307 ページの『有効な SECMAINT 特権または権限のリスト』
- 301 ページの『監査レコード・オブジェクト・タイプ』

有効な SECMAINT 特権または権限のリスト

下記は、有効な SECMAINT 特権または権限のリストです。

0x0000000000000001 Control Table
表またはビューに関して付与または取り消されたコントロール特権。

0x0000000000000002 ALTER TABLE
表を変更するために付与または取り消された特権。

0x0000000000000004 ALTER TABLE with GRANT
特権の付与が許可されている表を変更するために付与または取り消された特権。

0x0000000000000008 DELETE TABLE
表またはビューをドロップするために付与または取り消された特権。

0x0000000000000010 DELETE TABLE with GRANT
特権の付与が許可されている表をドロップするために付与または取り消された特権。

0x0000000000000020 Table Index
索引に関して付与または取り消された特権。

0x0000000000000040 Table Index with GRANT
特権の付与が許可されている索引に関して付与または取り消された特権。

0x0000000000000080 Table INSERT
表またはビューへの挿入に関して付与または取り消された特権。

0x0000000000000100 Table INSERT with GRANT
特権の付与が許可されている表への挿入に関して付与または取り消された特権。

0x0000000000000200 Table SELECT
表での選択に関して付与または取り消された特権。

0x0000000000000400 Table SELECT with GRANT
特権の付与が許可されている表での選択に関して付与または取り消された特権。

0x0000000000000800 Table UPDATE
表またはビューの更新に関して付与または取り消された特権。

0x0000000000001000 Table UPDATE with GRANT
特権の付与が許可されている表またはビューの更新に関して付与または取り消された特権。

0x0000000000002000 Table REFERENCE
表への参照に関して付与または取り消された特権。

0x0000000000004000 Table REFERENCE with GRANT
特権の付与が許可されている表への参照に関して付与または取り消された特権。

0x0000000000020000 CREATEIN Schema
スキーマに関して付与または取り消された CREATEIN 特権。

0x0000000000040000 CREATEIN Schema with GRANT
特権の付与が許可されているスキーマに関して付与または取り消された CREATEIN 特権。

0x0000000000080000 DROPIN Schema
スキーマに関して付与または取り消された DROPIN 特権。

0x0000000000100000 DROPIN Schema with GRANT
特権の付与が許可されているスキーマに関して付与または取り消された DROPIN 特権。

0x0000000000200000 ALTERIN Schema
スキーマに関して付与または取り消された ALTERIN 特権。

0x0000000000400000 ALTERIN Schema with GRANT
特権の付与が許可されているスキーマに関して付与または取り消された ALTERIN 特権。

0x0000000000800000 DBADM Authority
付与または取り消された DBADM 権限。

0x0000000001000000 CREATETAB Authority
付与または取り消された CREATETAB 権限。

0x0000000002000000 BINDADD Authority
付与または取り消された BINDADD 権限。

0x0000000004000000 CONNECT Authority
付与または取り消された CONNECT 権限。

0x0000000008000000 Create not fenced Authority
付与または取り消された fenced でないの作成権限。

0x0000000010000000 Implicit Schema Authority
付与または取り消された暗黙的スキーマ権限。

0x0000000020000000 Server PASSTHRU
このサーバー (フェデレーテッド・データベースのデータ・ソース) でパススルー機能を使用するために、付与または取り消された特権。

0x0000000100000000 Table Space USE
表スペースに表を作成するために付与または取り消された特権。

0x0000000200000000 Table Space USE with GRANT
特権の付与が許可されている表スペースに表を作成するために付与または取り消された特権。

0x0000000400000000 Column UPDATE
表の 1 つ以上の特定の列への更新に関して付与または取り消された特権。

0x0000000800000000 Column UPDATE with GRANT
特権の付与が許可されている表の 1 つ以上の特定の列への更新に関して付与または取り消された特権。

0x0000001000000000 Column REFERENCE
表の 1 つ以上の特定の列への参照に関して付与または取り消された特権。

0x0000002000000000 Column REFERENCE with GRANT
特権の付与が許可されている表の 1 つ以上の特定の列への参照に関して付与または取り消された特権。

0x0000004000000000 LOAD Authority
付与または取り消された LOAD 権限。

0x0000008000000000 Package BIND
パッケージに関して付与または取り消された BIND 特権。

0x0000010000000000 Package BIND with GRANT
特権の付与が許可されているパッケージに関して付与または取り消された BIND 特権。

0x0000020000000000 EXECUTE
パッケージに関して付与または取り消された EXECUTE 特権。

0x0000040000000000 EXECUTE with GRANT
特権の付与が許可されているパッケージまたはルーチンに関して付与または取り消された EXECUTE 特権。

0x0000080000000000 EXECUTE IN SCHEMA
スキーマ内のすべてのルーチンに関して付与または取り消された EXECUTE 特権。

0x0000100000000000 EXECUTE IN SCHEMA with GRANT
特権の付与が許可されているスキーマ内のすべてのルーチンに関して付与または取り消された EXECUTE 特権。

0x0000020000000000 EXECUTE IN TYPE
タイプ内のすべてのルーチンに関して付与または取り消された EXECUTE 特権。

0x0000400000000000 EXECUTE IN TYPE with GRANT
特権の付与が許可されているタイプ内のすべてのルーチンに関して付与または取り消された EXECUTE 特権。

0x0000080000000000 CREATE EXTERNAL ROUTINE
付与または取り消された CREATE EXTERNAL ROUTINE 特権。

0x0001000000000000 QUIESCE_CONNECT
付与または取り消された QUIESCE_CONNECT 特権。

関連資料:

• 306 ページの『SECMAINT イベントの監査レコード設計』

SYSADMIN イベントの監査レコード設計

表 12. SYSADMIN イベントの監査レコード設計

```
timestamp=1998-06-24-11.54.04.129923;category=SYSADMIN;audit event=DB2AUDIT;
  event correlator=1;event status=0;
  userid=boss;authid=BOSS;
  application id=*LOCAL.boss.980624155404;application name=db2audit;
```

NAME	フォーマット	記述
Timestamp	CHAR(26)	監査イベントの日付と時刻。
Category	CHAR(8)	監査イベントの区分。可能な値は以下のとおりです。 SYSADMIN
Audit Event	VARCHAR(32)	特定の監査イベント。 有効な値を、この表に続くリストに示しています。
Event Correlator	INTEGER	監査対象の操作の相関 ID。単一イベントにどの監査レコードが関連しているかを識別するために使用できます。
Event Status	INTEGER	監査イベントの状況、次のような 1 つの SQLCODE で表されます。 成功イベント >= 0 失敗イベント < 0
Database Name	CHAR(8)	どのイベントが生成されたかを示すデータベース名。インスタンス・レベルの監査イベントであった場合にはブランクとなります。
User ID	VARCHAR(1024)	監査イベントの時刻でのユーザー ID。
Authorization ID	VARCHAR(128)	監査イベントの時刻での許可 ID。
Origin Node Number	SMALLINT	監査イベントが発生したデータベース・パーティション番号。
Coordinator Node Number	SMALLINT	コーディネーター・データベース・パーティションのデータベース・パーティション番号。
Application ID	VARCHAR (255)	監査イベントが発生した時刻で使用していたアプリケーション ID。
Application Name	VARCHAR (1024)	監査イベントが発生した時刻で使用していたアプリケーション名。
Package Schema	VARCHAR (128)	監査イベントの時刻で使用していたパッケージのスキーマ。
Package Name	VARCHAR (128)	監査イベントが発生した時刻で使用していたパッケージ名。

表 12. SYSADMIN イベントの監査レコード設計 (続き)

timestamp=1998-06-24-11.54.04.129923;**category=SYSADMIN;audit event=DB2AUDIT;**
event correlator=1;event status=0;
userid=boss;authid=BOSS;
application id=*LOCAL.boss.980624155404;application name=db2audit;		

NAME	**フォーマット**	**記述**
Package Section Number	SMALLINT	監査イベントが発生した時刻で使用されていたパッケージのセクション番号。
Package Version	VARCHAR (64)	監査イベントが発生した時刻で使用していたパッケージのバージョン。

関連概念:

- 298 ページの『監査機能のレコード設計の紹介』

関連資料:

- 311 ページの『有効な SYSADMIN 監査イベントのリスト』

有効な SYSADMIN 監査イベントのリスト

下記は、有効な SYSADMIN 監査イベントのリストです。

表 13. SYSADMIN 監査イベント

START_DB2
STOP_DB2
CREATE_DATABASE
DROP_DATABASE
UPDATE_DBM_CFG
UPDATE_DB_CFG
CREATE_TABLESPACE
DROP_TABLESPACE
ALTER_TABLESPACE
RENAME_TABLESPACE
CREATE_NODEGROUP
DROP_NODEGROUP
ALTER_NODEGROUP
CREATE_BUFFERPOOL
DROP_BUFFERPOOL
ALTER_BUFFERPOOL
CREATE_EVENT_MONITOR
DROP_EVENT_MONITOR
ENABLE_MULTIPAGE
MIGRATE_DB_DIR
DB2TRC
DB2SET
ACTIVATE_DB
ADD_NODE
BACKUP_DB
CATALOG_NODE
CATALOG_DB
CATALOG_DCS_DB
CHANGE_DB_COMMENT
DEACTIVATE_DB
DROP_NODE_VERIFY
FORCE_APPLICATION
GET_SNAPSHOT
LIST_DRDA_INDOUBT_TRANSACTIONS
MIGRATE_DB
RESET_ADMIN_CFG
RESET_DB_CFG
RESET_DBM_CFG
RESET_MONITOR
RESTORE_DB
ROLLFORWARD_DB
SET_RUNTIME_DEGREE
SET_TABLESPACE_CONTAINERS
UNCATALOG_DB
UNCATALOG_DCS_DB
UNCATALOG_NODE
UPDATE_ADMIN_CFG
UPDATE_MON_SWITCHES
LOAD_TABLE
DB2AUDIT
SET_APPL_PRIORITY
CREATE_DB_AT_NODE
KILLDBM
MIGRATE_SYSTEM_DIRECTORY
DB2REMOT
DB2AUD
MERGE_DBM_CONFIG_FILE
UPDATE_CLI_CONFIGURATION
OPEN_TABLESPACE_QUERY
SINGLE_TABLESPACE_QUERY
CLOSE_TABLESPACE_QUERY
FETCH_TABLESPACE
OPEN_CONTAINER_QUERY
FETCH_CONTAINER_QUERY
CLOSE_CONTAINER_QUERY
GET_TABLESPACE_STATISTICS
DESCRIBE_DATABASE
ESTIMATE_SNAPSHOT_SIZE
READ_ASYNC_LOG_RECORD
PRUNE_RECOVERY_HISTORY
UPDATE_RECOVERY_HISTORY
QUIESCE_TABLESPACE
UNLOAD_TABLE
UPDATE_DATABASE_VERSION
CREATE_INSTANCE
DELETE_INSTANCE
SET_EVENT_MONITOR
GRANT_DBADM
REVOKE_DBADM
GRANT_DB_AUTHORITIES
REVOKE_DB_AUTHORITIES
REDIST_NODEGROUP

関連資料:

• 310 ページの『SYSADMIN イベントの監査レコード設計』

VALIDATE イベントの監査レコード設計

表 14. VALIDATE イベントの監査レコード設計

timestamp=1998-06-24-08.42.11.527490;**category=VALIDATE;audit event=CHECK_GROUP_MEMBERSHIP;**
event correlator=2;event status=-1092;
database=FOO;userid=boss;authid=BOSS;**execution id=newton;**
application id=*LOCAL.newton.980624124210;application name=testapp;
auth type=SERVER;		

NAME	フォーマット	記述
Timestamp	CHAR(26)	監査イベントの日付と時刻。
Category	CHAR(8)	監査イベントの区分。可能な値は以下のとおりです。 VALIDATE
Audit Event	VARCHAR(32)	特定の監査イベント。 有効な値は、GET_GROUPS、GET_USERID、AUTHENTICATE_PASSWORD、および VALIDATE_USER です。
Event Correlator	INTEGER	監査対象の操作の相関 ID。単一イベントにどの監査レコードが関連しているかを識別するために使用できます。
Event Status	INTEGER	監査イベントの状況、次のような 1 つの SQLCODE で表されます。 成功イベント >= 0 失敗イベント < 0
Database Name	CHAR(8)	どのイベントが生成されたかを示すデータベース名。インスタンス・レベルの監査イベントであった場合にはブランクとなります。
User ID	VARCHAR(1024)	監査イベントの時刻でのユーザー ID。
Authorization ID	VARCHAR(128)	監査イベントの時刻での許可 ID。
Execution ID	VARCHAR(1024)	監査イベントの時刻で使用していた実行 ID。
Origin Node Number	SMALLINT	監査イベントが発生したデータベース・パーティション番号。
Coordinator Node Number	SMALLINT	コーディネーター・データベース・パーティションのデータベース・パーティション番号。
Application ID	VARCHAR (255)	監査イベントが発生した時刻で使用していたアプリケーション ID。
Application Name	VARCHAR (1024)	監査イベントが発生した時刻で使用していたアプリケーション名。
Authentication Type	VARCHAR (32)	監査イベントの時刻での認証タイプ。
Package Schema	VARCHAR (128)	監査イベントの時刻で使用していたパッケージのスキーマ。
Package Name	VARCHAR (128)	監査イベントが発生した時刻で使用していたパッケージ名。
Package Section Number	SMALLINT	監査イベントが発生した時刻で使用されていたパッケージのセクション番号。
Package Version	VARCHAR (64)	監査イベントが発生した時刻で使用していたパッケージのバージョン。
Plug-in Name	VARCHAR(*32*)	監査イベントが発生した時点で使用されていたプラグインの名前。

関連概念:

- 298 ページの『監査機能のレコード設計の紹介』

CONTEXT イベントの監査レコード設計

表 15. CONTEXT イベントの監査レコード設計

```
timestamp=1998-06-24-08.42.41.476840;category=CONTEXT;audit event=EXECUTE_IMMEDIATE;
  event correlator=3;
  database=FOO;userid=boss;authid=BOSS;
  application id=*LOCAL.newton.980624124210;application name=testapp;
  package schema=NULLID;package name=SQLC28A1;
  package section=203;text=create table audit(c1 char(10), c2 integer);
```

NAME	フォーマット	記述
Timestamp	CHAR(26)	監査イベントの日付と時刻。
Category	CHAR(8)	監査イベントの区分。可能な値は以下のとおりです。 CONTEXT
Audit Event	VARCHAR(32)	特定の監査イベント。 有効な値を、この表に続くリストに示しています。
Event Correlator	INTEGER	監査対象の操作の相関 ID。単一イベントにどの監査レコードが関連しているかを識別するために使用できます。
Database Name	CHAR(8)	どのイベントが生成されたかを示すデータベース名。インスタンス・レベルの監査イベントであった場合にはブランクとなります。
User ID	VARCHAR(1024)	監査イベントの時刻でのユーザー ID。
Authorization ID	VARCHAR(128)	監査イベントの時刻での許可 ID。
Origin Node Number	SMALLINT	監査イベントが発生したデータベース・パーティション番号。
Coordinator Node Number	SMALLINT	コーディネーター・データベース・パーティションのデータベース・パーティション番号。
Application ID	VARCHAR (255)	監査イベントが発生した時刻で使用していたアプリケーション ID。
Application Name	VARCHAR (1024)	監査イベントが発生した時刻で使用していたアプリケーション名。
Package Schema	VARCHAR (128)	監査イベントの時刻で使用していたパッケージのスキーマ。
Package Name	VARCHAR (128)	監査イベントが発生した時刻で使用していたパッケージ名。
Package Section Number	SMALLINT	監査イベントが発生した時刻で使用されていたパッケージのセクション番号。
Statement Text (ステートメント)	CLOB(2M)	適用できる場合には、SQL ステートメントのテキストです。 SQL ステートメントのテキストが使用可能でない場合、NULL となります。
Package Version	VARCHAR (64)	監査イベントが発生した時刻で使用していたパッケージのバージョン。

関連概念:

- 298 ページの『監査機能のレコード設計の紹介』

関連資料:

- 314 ページの『有効な CONTEXT 監査イベントのリスト』

有効な CONTEXT 監査イベントのリスト

下記は、有効な CONTEXT 監査イベントのリストです。

表 16. CONTEXT 監査イベント

CONNECT	SET_APPL_PRIORITY
CONNECT_RESET	RESET_DB_CFG
ATTACH	GET_DB_CFG
DETACH	GET_DFLT_CFG
DARI_START	UPDATE_DBM_CFG
DARI_STOP	SET_MONITOR
BACKUP_DB	GET_SNAPSHOT
RESTORE_DB	ESTIMATE_SNAPSHOT_SIZE
ROLLFORWARD_DB	RESET_MONITOR
OPEN_TABLESPACE_QUERY	OPEN_HISTORY_FILE
FETCH_TABLESPACE	CLOSE_HISTORY_FILE
CLOSE_TABLESPACE_QUERY	FETCH_HISTORY_FILE
OPEN_CONTAINER_QUERY	SET_RUNTIME_DEGREE
CLOSE_CONTAINER_QUERY	UPDATE_AUDIT
FETCH_CONTAINER_QUERY	DBM_CFG_OPERATION
SET_TABLESPACE_CONTAINERS	DISCOVER
GET_TABLESPACE_STATISTIC	OPEN_CURSOR
READ_ASYNC_LOG_RECORD	CLOSE_CURSOR
QUIESCE_TABLESPACE	FETCH_CURSOR
LOAD_TABLE	EXECUTE
UNLOAD_TABLE	EXECUTE_IMMEDIATE
UPDATE_RECOVERY_HISTORY	PREPARE
PRUNE_RECOVERY_HISTORY	DESCRIBE
SINGLE_TABLESPACE_QUERY	BIND
LOAD_MSG_FILE	REBIND
UNQUIESCE_TABLESPACE	RUNSTATS
ENABLE_MULTIPAGE	REORG
DESCRIBE_DATABASE	REDISTRIBUTE
DROP_DATABASE	COMMIT
CREATE_DATABASE	ROLLBACK
ADD_NODE	REQUEST_ROLLBACK
FORCE_APPLICATION	IMPLICIT_REBIND

関連資料:

- 314 ページの『CONTEXT イベントの監査レコード設計』

監査機能のヒントと技法

CHECKING イベントを操作しているとき、ほとんどの場合、監査レコードのオブジェクト・タイプのフィールドは、オブジェクトにアクセスしようとするユーザー ID が必要な特権または権限を持っているか検査する対象のオブジェクトとなります。たとえば、ユーザーが 1 列を追加することによって表を ALTER しようとする場合、 CHECKING イベントの監査レコードは、試みたアクセスが ″ALTER″ であり、検査されているオブジェクト・タイプが ″TABLE″ であったことを必ず表示します (注: 表特権であるため、検査される必要があるのは列でない)。

ただし、オブジェクトの CREATE、BIND、または削除をユーザー ID に許可するデータベース権限あるかどうか検査する場合には、データベースに対して検査が行

われますが、オブジェクト・タイプ・フィールドは (データベース自体ではなく) 作成、バインド、またはドロップする対象のオブジェクトを指定します。

表に索引を作成するとき、索引の作成特権が必要です。このため、 CHECKING イベントの監査レコードのアクセス試行タイプは「CREATE」ではなく「INDEX」になります。

既存のパッケージをバインドしているとき、パッケージの DROP に対して OBJMAINT イベントの監査レコードが作成されます。さらに、パッケージの新しいコピーの CREATE に対して別の OBJMAINT イベントの監査レコードが作成されます。

SQL データ定義言語 (DDL) は、正常にログに記録される OBJMAINT または SECMAINT イベントを生成する可能性があります。しかし、イベントのロギングの後、エラーが原因で ROLLBACK が発生するかもしれません。その場合、オブジェクトが作成されないか、 GRANT または REVOKE 操作が不完全になります。このような場合に、CONTEXT イベントの使用が重要となります。このような CONTEXT イベントの監査レコード (特にイベントを終了するステートメント) は、操作試行の完了状態を示します。

監査レコードを DB2® Universal Database (DB2 UDB) のリレーショナル表にロードするためにふさわしい区切り文字付き ASCII 形式に抽出しているとき、ステートメントのテキスト・フィールド内で区切り文字を明確に指定しなければなりません。そうするには、区切り文字付き ASCII ファイルを抽出するとき、以下のようにします。

```
db2audit extract delasc delimiter <load delimiter>
```

この *load delimiter* は、単一文字 (たとえば ") または 16 進数で示される 4 バイト・ストリング (たとえば ″0xff″) です。有効なコマンドの例は次のとおりです。

```
db2audit extract delasc
db2audit extract delasc delimiter !
db2audit extract delasc delimiter 0xff
```

抽出時に区切り文字としてデフォルトのロード区切り文字 (″″″) 以外を使用した場合、 LOAD コマンドでは MODIFIED BY オプションを使用する必要があります。区切り文字として使用される ″0xff″ を指定した LOAD コマンド例の一部分を次に示します。

```
db2 load from context.del of del modified by chardel0xff replace into ...
```

これにより、デフォルトのロード文字ストリング区切り文字、 ″0xff″ がオーバーライドされます。

関連概念:

- 298 ページの『監査機能のレコード設計の紹介』

関連資料:

- 285 ページの『監査機能の使用方法』

DB2 UDB 監査機能アクティビティーの制御

手順:

監査機能アクティビティーの制御の説明のために、次のような簡単なシナリオを示します。ユーザー *newton* が、表を接続しかつ作成する *testapp* というアプリケーションを実行します。下記のそれぞれの例では、この同じアプリケーションを使用するとします。

まず、次のような極端な例を示します。すべての成功および不成功の監査イベントを監査したい場合には、次のように監査機能を構成します。

```
db2audit configure scope all status both
```

注: これにより、すべて監査可能なイベントに関する監査レコードが作成されます。結果として、多くの監査レコードが監査ログに書き込まれ、データベース・マネージャーのパフォーマンスを低下させます。この極端なケースは、デモンストレーションの目的でここに示したに過ぎません。上記のコマンドを使って監査機能を構成することはお勧めしません。

(「db2audit start」を使用した) この構成によって監査機能を開始した後、 *testapp* アプリケーションを実行すると、次のレコードが生成されて監査ログに格納されます。このログから監査レコードを抽出すれば、アプリケーションが実行する 2 つの操作に関する以下のレコードが生成されたことを確認できます。

アクション
作成されるレコードのタイプ

CONNECT

```
timestamp=1998-06-24-08.42.10.555345;category=CONTEXT;
audit event=CONNECT;event correlator=2;database=FOO;
application id=*LOCAL.newton.980624124210;
application name=testapp;

timestamp=1998-06-24-08.42.10.944374;category=VALIDATE;
audit event=AUTHENTICATION;event correlator=2;event status=0;
database=FOO;userid=boss;authid=BOSS;execution id=newton;
application id=*LOCAL.newton.980624124210;application name=testapp;
auth type=SERVER;

timestamp=1998-06-24-08.42.11.527490;category=VALIDATE;
audit event=CHECK_GROUP_MEMBERSHIP;event correlator=2;
event status=-1092;database=FOO;userid=boss;authid=BOSS;
execution id=newton;application id=*LOCAL.newton.980624124210;
application name=testapp;auth type=SERVER;

timestamp=1998-06-24-08.42.11.561187;category=VALIDATE;
audit event=CHECK_GROUP_MEMBERSHIP;event correlator=2;
event status=-1092;database=FOO;userid=boss;authid=BOSS;
execution id=newton;application id=*LOCAL.newton.980624124210;
application name=testapp;auth type=SERVER;

timestamp=1998-06-24-08.42.11.594620;category=VALIDATE;
audit event=CHECK_GROUP_MEMBERSHIP;event correlator=2;
event status=-1092;database=FOO;userid=boss;authid=BOSS;
execution id=newton;application id=*LOCAL.newton.980624124210;
application name=testapp;auth type=SERVER;

timestamp=1998-06-24-08.42.11.622984;category=CHECKING;
```

```
audit event=CHECKING_OBJECT;event correlator=2;event status=0;
database=FOO;userid=boss;authid=BOSS;
application id=*LOCAL.newton.980624124210;application name=testapp;
object name=FOO;object type=DATABASE;access approval reason=DATABASE;
access attempted=CONNECT;

timestamp=1998-06-24-08.42.11.801554;category=CONTEXT;
audit event=COMMIT;event correlator=2;database=FOO;userid=boss;
authid=BOSS;application id=*LOCAL.newton.980624124210;
application name=testapp;

timestamp=1998-06-24-08.42.41.450975;category=CHECKING;
audit event=CHECKING_OBJECT;event correlator=2;event status=0;
database=FOO;userid=boss;authid=BOSS;
application id=*LOCAL.newton.980624124210;application name=testapp;
package schema=NULLID;package name=SQLC28A1;object schema=NULLID;
object name=SQLC28A1;object type=PACKAGE;
access approval reason=OBJECT;access attempted=EXECUTE;
```

CREATE TABLE

```
timestamp=1998-06-24-08.42.41.476840;category=CONTEXT;
audit event=EXECUTE_IMMEDIATE;event correlator=3;database=FOO;
userid=boss;authid=BOSS;application id=*LOCAL.newton.980624124210;
application name=testapp;package schema=NULLID;package name=SQLC28A1;
package section=203;text=create table audit(c1 char(10), c2 integer);

timestamp=1998-06-24-08.42.41.539692;category=CHECKING;
audit event=CHECKING_OBJECT;event correlator=3;event status=0;
database=FOO;userid=boss;authid=BOSS;
application id=*LOCAL.newton.980624124210;application name=testapp;
package schema=NULLID;package name=SQLC28A1;package section=0;
object schema=BOSS;object name=AUDIT;object type=TABLE;
access approval reason=DATABASE;access attempted=CREATE;

timestamp=1998-06-24-08.42.41.570876;category=CHECKING;
audit event=CHECKING_OBJECT;event correlator=3;event status=0;
database=FOO;userid=boss;authid=BOSS;
application id=*LOCAL.newton.980624124210;application name=testapp;
package schema=NULLID;package name=SQLC28A1;package section=0;
object name=BOSS;object type=SCHEMA;access approval reason=DATABASE;
access attempted=CREATE;

timestamp=1998-06-24-08.42.41.957524;category=OBJMAINT;
audit event=CREATE_OBJECT;event correlator=3;event status=0;
database=FOO;userid=boss;authid=BOSS;
application id=*LOCAL.newton.980624124210;application name=testapp;
package schema=NULLID;package name=SQLC28A1;package section=0;
object schema=BOSS;object name=AUDIT;object type=TABLE;

timestamp=1998-06-24-08.42.42.018900;category=CONTEXT;
audit event=COMMIT;event correlator=3;database=FOO;userid=boss;
authid=BOSS;application id=*LOCAL.newton.980624124210;
application name=testapp;package schema=NULLID;
package name=SQLC28A1;
```

上記からわかるように、可能なすべての監査イベントおよびタイプの監査を要求する監査構成によって、非常に多くの監査レコードが生成されます。

ほとんどの場合、監査機能を構成するときには、監査対象のイベントを制限、つまり絞り込みます。たとえば、失敗したイベントだけを監査したい場合もあります。この場合、監査機能は次のように構成されます。

```
db2audit configure scope audit,checking,objmaint,secmaint,sysadmin,
   validate status failure
```

注: この構成は、初期の監査構成、つまり監査構成がリセットされた時点の構成です。

この構成を使って監査機能を開始し、 *testapp* アプリケーションを実行した後、次のレコードが生成されて監査ログに格納されます。 (なお、*testapp* がまだ一度も実行されていないことを想定します。) このログから監査レコードを抽出すれば、アプリケーションが実行する 2 つの操作に関する以下のレコードが生成されたことを確認できます。

アクション
作成されるレコードのタイプ

CONNECT

```
timestamp=1998-06-24-08.42.11.527490;category=VALIDATE;
audit event=CHECK_GROUP_MEMBERSHIP;event correlator=2;
event status=-1092;database=FOO;userid=boss;authid=BOSS;
execution id=newton;application id=*LOCAL.newton.980624124210;
application name=testapp;auth type=SERVER;

timestamp=1998-06-24-08.42.11.561187;category=VALIDATE;
audit event=CHECK_GROUP_MEMBERSHIP;event correlator=2;
event status=-1092;database=FOO;userid=boss;authid=BOSS;
execution id=newton;application id=*LOCAL.newton.980624124210;
application name=testapp;auth type=SERVER;

timestamp=1998-06-24-08.42.11.594620;category=VALIDATE;
audit event=CHECK_GROUP_MEMBERSHIP;event correlator=2;
event status=-1092;database=FOO;userid=boss;authid=BOSS;
execution id=newton;application id=*LOCAL.newton.980624124210;
application name=testapp;auth type=SERVER;
```

CREATE TABLE

(なし)

可能なすべての監査イベント (CONTEXT を除く) ではなく、監査試行が失敗したときにのみ監査を要求するよう監査を構成すれば、生成される監査レコードはかなり少なくなります。監査構成を変更することによって、生成される監査レコードのタイプおよび性質を制御できます。

監査機能を使用すれば、監査対象のユーザーがオブジェクトに対する特権を正常に付与されたときに、監査レコードを作成することができます。この場合、次のように監査機能を構成できます。

```
db2audit configure scope checking status success
```

この構成を使って監査機能を開始し、 *testapp* アプリケーションを実行した後、次のレコードが生成されて監査ログに格納されます。 (なお、*testapp* がまだ一度も実行されていないことを想定します。) このログから監査レコードを抽出すれば、アプリケーションが実行する 2 つの操作に関する以下のレコードが生成されたことを確認できます。

アクション
作成されるレコードのタイプ

CONNECT

```
timestamp=1998-06-24-08.42.11.622984;category=CHECKING;
audit event=CHECKING_OBJECT;event correlator=2;event status=0;
database=FOO;userid=boss;authid=BOSS;
```

```
timestamp=1998-06-24-08.42.41.450975;category=CHECKING;
audit event=CHECKING_OBJECT;event correlator=2;event status=0;
database=FOO;userid=boss;authid=BOSS;
application id=*LOCAL.newton.980624124210;application name=testapp;
package schema=NULLID;package name=SQLC28A1;object schema=NULLID;
object name=SQLC28A1;object type=PACKAGE;
access approval reason=OBJECT;access attempted=EXECUTE;

timestamp=1998-06-24-08.42.41.539692;category=CHECKING;
audit event=CHECKING_OBJECT;event correlator=3;event status=0;
database=FOO;userid=boss;authid=BOSS;
application id=*LOCAL.newton.980624124210;application name=testapp;
package schema=NULLID;package name=SQLC28A1;package section=0;
object schema=BOSS;object name=AUDIT;object type=TABLE;
access approval reason=DATABASE;access attempted=CREATE;

timestamp=1998-06-24-08.42.41.570876;category=CHECKING;
audit event=CHECKING_OBJECT;event correlator=3;event status=0;
database=FOO;userid=boss;authid=BOSS;
application id=*LOCAL.newton.980624124210;application name=testapp;
package schema=NULLID;package name=SQLC28A1;package section=0;
object name=BOSS;object type=SCHEMA;access approval reason=DATABASE;
access attempted=CREATE;
```

CREATE TABLE

(なし)

関連概念:

- 298 ページの『監査機能のレコード設計の紹介』

関連資料:

- 285 ページの『監査機能の使用方法』

第 3 部 付録

付録 A. 命名規則への準拠

一般的な命名規則

すべてのオブジェクトとユーザーの命名には規則があります。これらの規則には、作業しているプラットフォームに特有のものもあります。たとえば、名前に大文字と小文字を使用することに関連した規則があります。

- UNIX® プラットフォームでは、名前は小文字でなければなりません。
- Windows® プラットフォームでは、名前は大文字でも、小文字でも、大小混合でも構いません。

特に指定がない限り、名前には以下の文字を含めることができます。

- A から Z までの文字。名前に使用されるとき、多くの場合 A から Z は小文字から大文字に変換されます。
- 0 から 9 の数字。
- ! % ( ) { } . - ^ ~ _ (アンダースコアー) @、#、$、およびスペース。
- \ (バックスラッシュ)。

名前の先頭に、数値または下線文字を使用することはできません。

表、ビュー、列、索引、または許可 ID の名前には、SQL 予約語を使用しないでください。

ご使用のオペレーティング・システム、および DB2® Universal Database (DB2 UDB) を操作している場所によって、異なる働きをする特殊文字が他にもあります。それらの文字は正常に機能する可能性がありますが、必ず機能するという保証はありません。データベース内のオブジェクトを命名する際には、これら他の特殊文字を使用することはお奨めしません。

他にも、オブジェクト命名規則、ワークステーション命名規則、NLS 環境でのワークステーション命名規則、および Unicode 環境での命名規則を考慮する必要があります。

関連概念:

- 323 ページの『DB2 UDB オブジェクト命名規則』
- 328 ページの『ワークステーション命名規則』
- 326 ページの『ユーザー、ユーザー ID、およびグループの命名規則』
- 327 ページの『フェデレーテッド・データベース・オブジェクトの命名規則』

DB2 UDB オブジェクト命名規則

すべてのオブジェクトは、一般的な命名規則に従います。さらに、いくつかのオブジェクトには、表に示されているような追加の制約事項があります。

表 17. データベース、データベース別名、およびインスタンスの命名規則

オブジェクト	指針
• データベース • データベース別名 • インスタンス	• データベース名は、それらがカタログされるロケーションにおいてユニークでなければなりません。このロケーションとは、 DB2® Universal Database (DB2 UDB) の UNIX® ベースのインプリメンテーションではディレクトリー・パス、 Windows® インプリメンテーションでは論理ディスクです。 • データベース別名は、システム・データベース・ディレクトリー内でユニークでなければなりません。新規データベースが作成されると、デフォルトでは別名としてデータベース名が設定されます。そのため、たとえその名前のデータベースが存在しないとしても、データベース別名として既に存在する名前を使ってデータベースを作成することはできません。 • データベース、データベース別名、およびインスタンス名の最大長は 8 バイトです。 • Windows NT®、Windows 2000、Windows XP、および Windows Server 2003 システムでは、インスタンスはサービス名と同じ名前を持つことができません。 **注:** 通信環境でデータベースを使用する場合、問題を未然に防ぐために、特殊文字 @、#、および $ はデータベース名に使用しないでください。さらに、これらの特殊文字はすべてのキーボードに共通ではないので、他の言語でデータベースを使用する場合にも使用しないでください。

表 18. データベース・オブジェクトの命名規則

オブジェクト	指針
• 別名 • バッファー・プール • 列 • イベント・モニター • 索引 • メソッド • ノード・グループ • パッケージ • パッケージ・バージョン • スキーマ • ストアード・プロシージャー • 表 • 表スペース • トリガー • UDF • UDT • ビュー	使用できるバイト数は最大 18 バイトです。ただし、次の名前は*例外* です。 • 表名 (ビュー名、サマリー表名、別名、および相関名)。128 バイトまで指定できます。 • 列名は 30 バイトまで指定できます。 • パッケージ名。8 バイトまで指定できます。 • スキーマ名。30 バイトまで指定できます。 • パッケージ・バージョン。64 バイトまで指定できます。 • オブジェクト名。次のものを含めることができます。 – 有効なアクセント付き文字 (ö など) – マルチバイト文字。マルチバイト・スペースは除く (マルチバイト環境の場合) • パッケージ名およびパッケージ・バージョンにはピリオド (.)、ハイフン (-)、およびコロン (:) もまた含めることができます。

表 19. フェデレーテッド・データベース・オブジェクトの命名規則

オブジェクト	指針
• 関数マッピング • 索引の指定 • ニックネーム • サーバー • タイプ・マッピング • ユーザー・マッピング • ラッパー	• ニックネーム、マッピング、索引の指定、サーバー、およびラッパーの名前は 128 バイト以下でなければなりません。 • サーバーおよびニックネームのオプションおよびオプション設定値は 255 バイトに限定されています。 • フェデレーテッド・データベース・オブジェクトの名前には、次のものも含めることができます。 – 有効なアクセント付き文字 (ö など) – マルチバイト文字。マルチバイト・スペースは除く (マルチバイト環境の場合)

区切り ID およびオブジェクト名:

キーワードを使用することができます。キーワードが SQL キーワードとしても解釈されるコンテキストで使用する場合には、それを区切り ID として指定する必要があります。

区切り ID を使用することによって、上記の命名規則に違反するオブジェクトを作成することは可能ですが、そのオブジェクトを使おうとするとエラーになります。たとえば、名前に + または - 記号が含まれている列を作成し、その列を索引の列として使おうとすると、索引の表を認識する段階で問題が起きてしまいます。

スキーマ名の追加情報:

- ユーザー定義タイプ (UDT) には、8 バイトを超えるスキーマ名を指定することはできません。
- 次のスキーマ名は予約語なので、使用しないでください。SYSCAT、SYSFUN、SYSIBM、SYSSTAT。
- 移行の問題を未然に防ぐために、SYS で始まるスキーマ名を使用しないでください。データベース・マネージャーでは SYS で始まるスキーマ名を使用して、トリガー、ユーザー定義タイプ、またはユーザー定義関数を作成できません。
- SESSION をスキーマ名として使用しないようお奨めします。宣言される一時表は、SESSION によって修飾しなければなりません。したがって、アプリケーションに永続表と同じ名前の一時表を宣言させることができますが、その場合、アプリケーション・ロジックが複雑になりすぎる可能性があります。宣言される一時表を扱う場合以外は、スキーマ SESSION の使用を避けてください。

関連概念:

- 323 ページの『一般的な命名規則』

区切り ID およびオブジェクト名

キーワードを使用することができます。キーワードが SQL キーワードとしても解釈されるコンテキストで使用される場合には、それを区切り ID として指定する必要があります。

区切り ID を使用することによって、上記の命名規則に違反するオブジェクトを作成することは可能ですが、そのオブジェクトを使おうとするとエラーになります。

たとえば、名前に + または - 記号が含まれている列を作成し、その列を索引の列として使おうとすると、索引の表を認識する段階で問題が起きてしまいます。

関連概念:

- 323 ページの『一般的な命名規則』

ユーザー、ユーザー ID、およびグループの命名規則

表 20. ユーザー、ユーザー ID、およびグループの命名規則

オブジェクト	指針
• グループ名 • ユーザー名 • ユーザー ID	• グループ名は最大 30 文字です。 • Linux および UNIX® ベースのシステムでは、ユーザー ID は最大 8 文字です。 • Windows® 上のユーザー名は、最大 30 文字です。 Windows NT®、Windows 2000、Windows XP、および Windows Server 2003 では現在のところ、実際には 20 文字が限度となっています。 • クライアント認証を使用しない場合は、ユーザー名およびパスワードが明示的に指定されていれば、 Windows NT、Windows 2000、 Windows XP、および Windows Server 2003 に接続している、ユーザー名が 8 文字を超える Windows 以外の 32 ビット・クライアントがサポートされます。 • 名前および ID には、次の禁止事項があります。 – USERS、 ADMINS、 GUESTS、 PUBLIC、 LOCAL、または任意の SQL 予約語であってはならない。 – IBM®、SQL、または SYS で始めてはならない。 – アクセント付き文字を組み込まない。

注:

1. いくつかのオペレーティング・システムでは、大文字小文字の区別があるユーザー ID およびパスワードを使用できます。それが使用できるかどうかは、ご使用のオペレーティング・システムの資料で確認してください。
2. 正常実行された CONNECT または ATTACH から戻された許可 ID は、8 文字で切り捨てられます。省略符号 (...) が許可 ID に追加され、 SQLWARN フィールドに切り捨てられたことを示す警告が入ります。
3. ユーザー ID およびパスワードの末尾ブランクは、除去されます。

関連概念:

- 323 ページの『一般的な命名規則』
- 327 ページの『フェデレーテッド・データベース・オブジェクトの命名規則』

フェデレーテッド・データベース・オブジェクトの命名規則

表 21. フェデレーテッド・データベース・オブジェクトの命名規則

オブジェクト	指針
• 関数マッピング • 索引の指定 • ニックネーム • サーバー • タイプ・マッピング • ユーザー・マッピング • ラッパー	• ニックネーム、マッピング、索引の指定、サーバー、およびラッパーの名前は 128 バイト以下でなければなりません。 • サーバーおよびニックネームのオプションおよびオプション設定値は 255 バイトに限定されています。 • フェデレーテッド・データベース・オブジェクトの名前には、次のものも含めることができます。 – 有効なアクセント付き文字 (ö など) – マルチバイト文字。マルチバイト・スペースは除く (マルチバイト環境の場合)

関連概念:

- 323 ページの『一般的な命名規則』

スキーマ名の使用に関する追加の制限および推奨事項

- ユーザー定義タイプ (UDT) には、8 バイトを超えるスキーマ名を指定することはできません。
- 次のスキーマ名は予約語なので、使用しないでください。SYSCAT、SYSFUN、SYSIBM、SYSSTAT。
- 移行の問題を未然に防ぐために、SYS で始まるスキーマ名を使用しないでください。データベース・マネージャーでは SYS で始まるスキーマ名を使用して、トリガー、ユーザー定義タイプ、またはユーザー定義関数を作成できません。
- SESSION をスキーマ名として使用しないようお奨めします。宣言される一時表は、SESSION によって修飾しなければなりません。したがって、アプリケーションに永続表と同じ名前の一時表を宣言させることができますが、その場合、アプリケーション・ロジックが複雑になりすぎる可能性があります。宣言される一時表を扱う場合以外は、スキーマ SESSION の使用を避けてください。

関連概念:

- 323 ページの『一般的な命名規則』

サーバー上でのパスワードの保守

パスワードのメンテナンス作業を実行する必要があるかもしれません。そのような作業はサーバーにおいて必要ですが、その際、サーバー環境で作業できないユーザーや快適に作業できないユーザーが多数生じるため、これらの作業の実行は簡単ではありません。 DB2® Universal Database (DB2 UDB) の提供する機能を使用すれば、サーバーにいなくても、パスワードを更新および確認することができます。たとえば、DB2 for OS/390® バージョン 5 では、この方法を使用してユーザー・パスワードを変更できます。エラー・メッセージ SQL1404N「パスワードの有効期限が切れています。」を受け取った場合は、次のように CONNECT ステートメントを使用してパスワードを変更します。

```
CONNECT TO <database> USER <userid> USING <password>
   NEW <new_password> CONFIRM <new_password>
```

また、DB2 UDB 構成アシスタント (CA) の「パスワードの変更 (Password change)」ダイアログを使用してパスワードを変更することもできます。

関連概念:

- 323 ページの『一般的な命名規則』
- 323 ページの『DB2 UDB オブジェクト命名規則』
- 328 ページの『ワークステーション命名規則』
- 326 ページの『ユーザー、ユーザー ID、およびグループの命名規則』
- 327 ページの『フェデレーテッド・データベース・オブジェクトの命名規則』
- 325 ページの『区切り ID およびオブジェクト名』
- 327 ページの『スキーマ名の使用に関する追加の制限および推奨事項』

ワークステーション命名規則

ワークステーション名 は、ローカル・ワークステーションに常駐するデータベース・サーバー、データベース・クライアント、または DB2® Universal Database (DB2 UDB) Personal Edition の NetBIOS 名を指定します。この名前は、データベース・マネージャー構成ファイルに保管されます。ワークステーション名はワークステーション *nname* ともいいます。

さらに、指定する名前には、以下の条件があります。

- 指定できる桁数は 1〜8 文字です。
- &、#、または @ を入れることはできません。
- ネットワーク内で必ずユニークでなければなりません。

パーティション・データベース・システムでは、パーティション・データベース・システム全体を表す、ワークステーション *nname* が 1 つだけありますが、各ノードにはそれぞれ派生したユニークな NetBIOS *nname* があります。

パーティション・データベース・システムを表すワークステーション *nname* は、インスタンスを所有するデータベース・パーティション・サーバーのデータベース・マネージャー構成ファイルに保管されます。

各ノードのユニークな *nname* は、ワークステーション *nname* およびノード番号から派生されます。

ノードがインスタンスを所有していない場合には、 NetBIOS *nname* は次のように派生されます。

1. インスタンスが所有するマシンのワークステーション *nname* の先頭文字が、ノードの NetBIOS *nname* の先頭文字として使用されます。
2. 次の 1〜3 文字はノード番号を表します。範囲は 1〜999 です。
3. 残りの文字は、インスタンスが所有するマシンのワークステーション *nname* から取られます。残りの文字の数は、インスタンスが所有するマシンのワークステーション *nname* の長さによって異なります。この番号の範囲は、0〜4 です。

たとえば、以下のとおりです。

インスタンスが所有するマシンの *ワークステーション nname*	ノード番号	導出されるノード NetBIOS *nname*
GEORGE	3	G3ORGE
A	7	A7
B2	94	B942
N0076543	21	N216543
GEORGE5	1	G1RGE5

デフォルトのワークステーション *nname* をインストール中に変更した場合、ワークステーション *nname* の最後の 4 文字は、 NetBIOS *nname* の対立の原因が生じる可能性を最小限にするため、 NetBIOS ネットワーク全体でユニークなものでなければなりません。

関連概念:

- 323 ページの『一般的な命名規則』

NLS 環境での命名規則

データベース名の中で使用できる基本文字セットは、単一バイトの大文字および小文字のローマ字 (A...Z、a...z)、アラビア数字 (0...9) および下線文字 (_) から構成されます。ホストのデータベース製品との互換性を保つために、 3 つの特殊文字 (#、@、および $) がこのリストに加えられています。特殊文字 #、@、および $ は NLS ホスト (EBCDIC) 不変文字セットに組み込まれていないため、これらを NLS 環境で使用する場合は注意してください。使用されているコード・ページに応じて、拡張文字セットの文字も使用できます。複数コード・ページ環境でデータベースを使用している場合、使用する拡張文字セットのどのエレメントもすべてのコード・ページによってサポートされていることを確認する必要があります。

データベース・オブジェクト (表やビューなど) に命名するときは、プログラム・ラベル、ホスト変数、カーソル、および拡張文字セットからのエレメント (たとえば、発音区別符付きの文字) も使用することができます。厳密にどの文字を使用できるかは、使用しているコード・ページによって異なります。

DBCS ID 用の拡張文字セットの定義:

DBCS 環境では、拡張文字セットは、基本文字セットにあるすべての文字、および次のような文字から構成されます。

- それぞれの DBCS コード・ページにある 2 バイト文字は、すべて (2 バイトのスペースは除く) 有効な文字です。
- 2 バイトのスペースは、特殊文字です。
- それぞれの混合コード・ページで使用できる単一バイト文字は、次のように様々なカテゴリーに割り当てられます。

カテゴリー	各混合コード・ページにある有効なコード・ポイント
数字	x30-39

カテゴリー	各混合コード・ページにある有効なコード・ポイント
文字	x23-24、x40-5A、x61-7A、 xA6-DF (コード・ページ 932 および 942 の場合のみ A6-DF)
特殊文字	その他のすべての有効な単一バイト文字のコード・ポイント

関連概念:

- 323 ページの『一般的な命名規則』
- 323 ページの『DB2 UDB オブジェクト命名規則』
- 328 ページの『ワークステーション命名規則』

Unicode 環境での命名規則

UCS-2 データベースでは、ID はすべてマルチバイトの UTF-8 です。したがって、拡張文字セット (たとえば、アクセント付き文字、マルチバイト文字) の文字の使用が DB2® Universal Database (DB2 UDB) によって許可されている場合には、 ID として任意の UCS-2 文字を使用できます。

クライアントは、それぞれの環境でサポートされている任意の文字を入力することができます。その後、ID に入れられたすべての文字がデータベース・マネージャーによって UTF-8 へ変換されます。 UCS-2 データベースで ID に各国語の文字を指定するときには、以下の 2 つの点を考慮する必要があります。

- 非 ASCII 文字にはそれぞれ 2 から 4 バイトが必要。そのため、*n* バイトの ID は、ASCII 文字と非 ASCII 文字の比率に応じて、 *n*/4 文字から *n* 文字の範囲を保持できます。非 ASCII 文字 (アクセントなど) が 1 文字か 2 文字しか含まれない場合、その上限は *n* 文字に近くなりますが、完全に非 ASCII である ID の場合 (たとえば、日本語の場合)、使用できるのは *n*/4 から *n*/3 文字だけです。
- ID を別のクライアント環境から入力するのであれば、そのクライアントで使用できる文字の共通サブセットを使用して、 ID を定義しなければならない。たとえば、UCS-2 データベースが Latin-1、アラビア語、および日本語環境からアクセスされる場合、すべての ID を実際に ASCII に限定する必要があります。

関連概念:

- 323 ページの『一般的な命名規則』
- 323 ページの『DB2 UDB オブジェクト命名規則』
- 328 ページの『ワークステーション命名規則』

付録 B. クライアントの自動転送の使用

この付録では、クライアントの自動転送について紹介し、ご使用の環境でこれを正しく機能させる方法を示します。

クライアントの自動転送についての説明およびセットアップ

サーバー・クラッシュが発生すると、そのサーバーに接続している各クライアントは通信エラーを受け取り、接続が終了してアプリケーション・エラーが発生します。可用性を保つ必要が大きい場合には、通常、重複セットアップ、またはサーバーをスタンバイ・ノードにフェイルオーバーする機能をインプリメントします。どちらの場合も、DB2® Universal Database (DB2 UDB) クライアント・コードは、フェイルオーバー・ノードで稼働している可能性のある元のサーバーとの接続を再確立するか (IP アドレスもまたフェイルオーバーします)、新しいサーバーとの接続を再確立しようとします。

接続が再確立されると、アプリケーションはトランザクション失敗を通知するエラー・メッセージを受け取りますが、アプリケーションはその次のトランザクションを続行できます。

クライアント自動転送機能の主な目的は、 DB2 UDB クライアント・アプリケーションが通信の障害から回復し、中断を最小限に抑えて機能を続行できるようにすることです。その名前が示すように、連続稼働をサポートするうえで転送は中心的な役割を果たします。ただし、クライアント接続が識別できるような ALTERNATE SERVER の場所が存在する場合に限って、転送が可能になります。

クライアント自動転送機能は、たとえば以下のような構成可能な環境で使用できます。

1. Enterprise Server Edition (ESE) - データ・パーティション機能 (DPF) を使用
2. Data Propagator (DPROPR) 型の複製
3. High Availability Cluster Multiprocessor (HACMP)
4. 高可用性災害時リカバリー (HADR)

   クライアント自動転送が HADR と連動することにより、クライアント・アプリケーションは、機能の中断を最小限に抑えて、アクセス先データベースのフェイルオーバー後に稼働を続けることができます。

DB2 UDB が通信障害から回復できるようにするためには、通信障害が発生する前に、ALTERNATE SERVER の場所を指定しておく必要があります。 UPDATE ALTERNATE SERVER FOR DATABASE コマンドを使用して、 ALTERNATE SERVER を指定できます。 ALTERNATE SERVER の場所をすべてのクライアントに適用させるためには、サーバー・サイドで ALTERNATE SERVER の場所を指定する必要があります。 ALTERNATE SERVER をクライアント・インスタンスで設定した場合、それは無視されます。

たとえば、「N1」というノード (ホスト名 XXX、ポート番号 YYY) にデータベースが存在するとします。データベース管理者は、ALTERNATE SERVER の場所をホスト名 AAA、ポート番号 123 に設定することを決定します。データベース管理者は、ノード N1 (サーバー・インスタンス上) で以下のようなコマンドを実行することができます。

```
db2 update alternate server for database db2 using hostname AAA port 123
```

データベース管理者がサーバー・インスタンスの特定のデータベース上で ALTERNATE SERVER の場所を指定した後、 ALTERNATE SERVER の場所は、接続時にクライアントに戻されます。何らかの理由で通信が失われた場合、DB2 UDB クライアント・コードは、サーバーから戻された ALTERNATE SERVER 情報を使って接続を再確立できるようになります。サーバーにおいて ALTERNATE SERVER が指定されない場合、クライアント自動転送は処理されません。

ALTERNATE SERVER が指定されている場合、一般的に、通信エラー (sqlcode -30081) または sqlcode -1224 が検出されたときにクライアント自動転送が使用可能になります。ただし、高可用性災害時リカバリー (HADR) 環境では、 sqlcode -1776 が HADR スタンバイ・サーバーから戻された場合にもまた、これが使用可能になります。

関連概念:

- 332 ページの『クライアント自動転送の制約事項』

関連資料:

- 334 ページの『クライアント自動転送の例』

クライアント自動転送の制約事項

クライアント自動転送機能を使用するとき、いくつかの制約事項があります。

- クライアント自動転送は、 DB2® Universal Database (DB2 UDB) サーバーまたは DB2 Connect™ サーバーへの接続に使用される通信プロトコルが TCP/IP の場合にのみ、サポートされます。つまり、TCP/IP 以外のプロトコルを使って接続している場合、クライアント自動転送機能は使用可能になりません。たとえ DB2 UDB でループバックがセットアップされている場合でも、クライアント自動転送機能を利用するためには、 TCP/IP 通信プロトコルを使用する必要があります。
- ALTERNATE SERVER の場所への通信が再確立された場合、同じデータベース別名への新しい接続はすべて、 ALTERNATE SERVER の場所に接続されます。元のロケーションの問題が修正され、新しい接続を元のロケーションとの間で確立したい場合には、以下のいくつかのオプションから選択できます。
  - ALTERNATE SERVER をオフラインにして、接続が元のサーバーに再びフェイルオーバーするようにします。 (この場合、元のサーバーを ALTERNATE SERVER の ALTERNATE SERVER の場所として設定するために、 UPDATE ALTERNATE SERVER コマンドを使って元のサーバーがすでにカタログされていることを想定します。)
  - 新しい接続によって使用される新しいデータベース別名をカタログすることができます。
  - データベース項目をアンカタログして、再びカタログすることができます。

- Linux、UNIX®、および Windows® オペレーティング・システム用の DB2 UDB は、クライアント・サイドとサーバー・サイドの両方でクライアント自動転送機能をサポートします。その他の DB2 UDB ファミリーは、現時点では、この機能をサポートしません。
- DB2 UDB for z/OS™ でのデータ共用 SYSPLEX セットアップにより、接続先として可能な複数のサーバーのリストを戻すことができます。ただし、通信障害が発生した場合、このリストはメモリーにのみ保持されます。そのとき、DB2 UDB クライアントはリストを使用して、接続先として適切な ALTERNATE SERVER の場所を判別します。
- ALTERNATE SERVER にインストールされた DB2 UDB は、元のサーバーにインストールされた DB2 UDB と比べて、同じバージョンでなければなりませんが、フィックスパック・レベルは高くすることができます。
- クライアント・マシンでデータベース・ディレクトリーを更新する権限があるかどうかにかかわらず、 ALTERNATE SERVER 情報は常にメモリーに保持されます。言い換えると、データベース・ディレクトリーの更新権限がない場合 (または、読み取り専用のデータベース・ディレクトリーである場合)、他のサーバーとの間でメモリーが共用されないため、他のアプリケーションは ALTERNATE SERVER を判別して使用することができません。
- すべての代替ロケーションの間で、同じ認証が適用されます。つまり、代替ロケーションの認証タイプが元のロケーションと異なる場合、クライアントはデータベース接続を再確立できません。
- 通信障害が発生した場合、グローバル一時表、ID、シーケンス、サーバー、フェデレート処理用のサーバー・オプション (SET SERVER OPTION)、および特殊レジスターなど、すべてのセッション・リソースが失われます。処理を続行するためには、アプリケーションがセッション・リソースを再確立しなければなりません。接続が再確立された後、特殊レジスター・ステートメントを実行する必要はありません。これは、通信エラーの前に発行された特殊レジスター・ステートメントを DB2 UDB が再生するためです。ただし、一部の特殊レジスターは再生されません。以下のものが該当します。
  - SET ENCRYPTPW
  - SET EVENT MONITOR STATE
  - SET SESSION AUTHORIZATION
  - SET TRANSFORM GROUP

  注: クライアントが CLI、JCC Type 2 または Type 4 ドライバーを使用している場合、接続が再確立された後、元のサーバーに対して準備された SQL ステートメントは新しいサーバーに対して暗黙的に再び準備されます。ただし、組み込み SQL ルーチン (たとえば SQC または SQX アプリケーション) は再び準備されません。

クライアント自動転送を実行する別の方法は、DNS 項目を使用して、 DNS 項目用の代替 IP アドレスを指定することです。この基本にある考え方は、 2 番目の IP アドレス (ALTERNATE SERVER の場所) を DNS 項目に指定することです。クライアントは ALTERNATE SERVER を認識しませんが、接続時に DB2 UDB が DNS 項目用の複数の IP アドレス間を切り替えます。

関連概念:

- 81 ページの『自動クライアント・リルートのインプリメンテーション』
- 331 ページの『クライアントの自動転送についての説明およびセットアップ』

関連タスク:

- 77 ページの『データベースの代替サーバーを識別する』

関連資料:

- 334 ページの『クライアント自動転送の例』

クライアント自動転送の例

以下は、クライアント・アプリケーションでのクライアント自動転送の例です (疑似コードだけを使用しています)。

```
int checkpoint = 0;

check_sqlca(unsigned char *str, struct sqlca *sqlca)
{
   if (sqlca->sqlcode == -30081)
   {
      // as communication is lost, terminate the application right away
      exit(1);
   }
   else
   {
      // print out the error
      printf(...);

 if (sqlca->sqlcode == -30108)
 {
    // connection is re-established, re-execute the failed transaction
         if (checkpoint == 0)
         {
             goto checkpt0;
         }
    else if (checkpoint == 1)
         {
            goto checkpt1;
         }
         else if (checkpoint == 2)
         {
             goto checkpt2;
         }
         ....
         exit;
 }
   }
}

main()
{
   connect to mydb;
   check_sqlca("connect failed", &sqlca);

checkpt0:
   EXEC SQL set current schema XXX;
   check_sqlca("set current schema XXX failed", &sqlca);

   EXEC SQL create table t1...;
   check_sqlca("create table t1 failed", &sqlca);

   EXEC SQL commit;
   check_sqlca("commit failed", &sqlca);
```

```
    if (sqlca.sqlcode == 0)
    {
       checkpoint = 1;
    }

checkpt1:
    EXEC SQL set current schema YYY;
    check_sqlca("set current schema YYY failed", &sqlca);

    EXEC SQL create table t2...;
    check_sqlca("create table t2 failed", &sqlca);

    EXEC SQL commit;
    check_sqlca("commit failed", &sqlca);

    if (sqlca.sqlcode == 0)
    {
       checkpoint = 2;
    }
...
}
```

クライアント・マシンでは、「mydb」というデータベースがカタログされてノード「hornet」を参照します。この「hornet」もまた、ノード・ディレクトリーでカタログされます (ホスト名「hornet」、ポート番号 456)。

例 1 (非 HADR データベースを使用する場合)

サーバー「hornet」(ホスト名およびポート番号は hornet と同じ) において、データベース「mydb」が作成されます。さらに、ALTERNATE SERVER (ホスト名「montero」、ポート番号 456) でもまた、データベース「mydb」が作成されます。それに加えて、サーバー「hornet」において、データベース「mydb」用の ALTERNATE SERVER を以下のように更新する必要があります。

```
db2 update alternate server for database mydb using hostname montero port 456
```

上記のサンプル・アプリケーションでは、クライアント自動転送機能をセットアップしない場合、 `create table t1` ステートメントで通信エラーが発生するとアプリケーションは終了します。クライアント自動転送機能をセットアップした場合、DB2 UDB はホスト「hornet」(ポート 456) への接続を再び確立しようとします。このホストがまだ非稼働中であれば、 DB2 UDB は ALTERNATE SERVER の場所 (ホスト「montero」、ポート 456) への接続を試行します。 ALTERNATE SERVER の場所との接続で通信エラーが発生しなければ、アプリケーションは後続のステートメントを実行し続ける (失敗したトランザクションを再実行する) ことができます。

例 2 (HADR データベースを使用する場合)

サーバー「hornet」(ホスト名およびポート番号は hornet と同じ) において、 1 次データベース「mydb」が作成されます。さらに、ホスト「montero」(ポート 456) において 2 次データベースが作成されます。 1 次および 2 次データベース用に HADR をセットアップする方法については、「*データ・リカバリーと高可用性 ガイドおよびリファレンス*」を参照してください。それに加えて、データベース「mydb」用の ALTERNATE SERVER を以下のように更新する必要があります。

```
db2 update alternate server for database mydb using hostname montero port 456
```

上記のサンプル・アプリケーションでは、クライアント自動転送機能をセットアップしない場合、 `create table t1` ステートメントで通信エラーが発生するとアプリケーションは終了します。クライアント自動転送機能をセットアップした場合、DB2 UDB はホスト「hornet」(ポート 456) への接続を再び確立しようとします。このホストがまだ非稼働中であれば、 DB2 UDB は ALTERNATE SERVER の場所 (ホスト「montero」、ポート 456) への接続を試行します。 ALTERNATE SERVER の場所との接続で通信エラーが発生しなければ、アプリケーションは後続のステートメントを実行し続ける (失敗したトランザクションを再実行する) ことができます。

関連概念:

- 331 ページの『クライアントの自動転送についての説明およびセットアップ』

関連タスク:

- 77 ページの『データベースの代替サーバーを識別する』

付録 C. Lightweight Directory Access Protocol (LDAP) ディレクトリー・サービスの使用

Lightweight Directory Access Protocol (LDAP) の紹介

Lightweight Directory Access Protocol (LDAP) は、ディレクトリー・サービスに対する業界標準のアクセス方式です。ディレクトリー・サービスとは、分散環境内にある複数のシステムおよびサービスについてのリソース情報を集めたリポジトリーです。クライアントとサーバーはディレクトリー・サービスを使用して、それらのリソースにアクセスします。各データベース・サーバーのインスタンスは自らの存在を LDAP サーバーに公表するとともに、データベースの作成時にはデータベース情報を LDAP ディレクトリーへ送信します。クライアントがデータベースに接続すると、 LDAP ディレクトリーからそのサーバーのカタログ情報を取り出せます。各クライアントは、それぞれのマシンでローカルにカタログ情報を保管する必要はなくなります。クライアント・アプリケーションは、LDAP ディレクトリーの中で、データベースへ接続するのに必要な情報を探します。

キャッシュ・メカニズムが備えられているので、クライアントは LDAP ディレクトリーを一度探すだけで済みます。 LDAP ディレクトリー・サーバーから情報を検索したら、 *dir_cache* データベース・マネージャー構成パラメーターおよび DB2LDAPCACHE レジストリー変数の値に基づいて、その情報はローカル・マシンに格納されるか、キャッシュに入れられます。データベース・マネージャー構成パラメーター *dir_cache* は、データベース、ノード、および DCS ディレクトリー・ファイルをメモリー・キャッシュに保管するために使用されます。ディレクトリー・キャッシュは、アプリケーションがクローズするまでアプリケーションによって使用されます。レジストリー変数 DB2LDAPCACHE は、データベース、ノード、および DCS ディレクトリー・ファイルをローカル・ディスク・キャッシュに保管するために使用されます。

- DB2LDAPCACHE=NO かつ *dir_cache*=NO の場合、情報は必ず LDAP から読み取られます。
- DB2LDAPCACHE=NO かつ *dir_cache*=YES の場合、 LDAP から情報が一度読み取られ、その情報は DB2® キャッシュに挿入されます。
- DB2LDAPCACHE=YES であるか、これが設定されない場合には、 LDAP から情報が一度読み取られ、その情報はローカル・データベース、ノード、および DCS ディレクトリーにキャッシュされます。

注: DB2LDAPCACHE レジストリー変数が適用されるのは、データベースとノード・ディレクトリーだけです。

関連概念:

- 79 ページの『Lightweight Directory Access Protocol (LDAP) ディレクトリー・サービス』
- 339 ページの『Active Directory のサポート』
- 355 ページの『LDAP サポートと DB2 Connect』

- 356 ページの『LDAP 環境でのセキュリティーに関する考慮事項』
- 357 ページの『Active Directory のセキュリティーに関する考慮事項』
- 「*管理ガイド: パフォーマンス*」の『DB2 レジストリー変数と環境変数』
- 358 ページの『DB2 オブジェクト・クラスおよび属性を使用して LDAP ディレクトリー・スキーマを拡張する』

関連タスク:

- 340 ページの『Active Directory を使用するよう DB2 を構成する』
- 341 ページの『IBM LDAP 環境での DB2 の構成』
- 342 ページの『LDAP ユーザーの作成』
- 343 ページの『DB2 アプリケーション用 LDAP ユーザーの構成』
- 343 ページの『インストール後の DB2 サーバーの登録』
- 345 ページの『DB2 サーバー用のプロトコル情報を更新する』
- 347 ページの『アタッチのためにノード別名をカタログする』
- 347 ページの『DB2 サーバーの登録を解除する』
- 348 ページの『LDAP ディレクトリーへのデータベースの登録』
- 348 ページの『LDAP 環境でのリモート・サーバーのアタッチ』
- 349 ページの『LDAP ディレクトリーからのデータベースの登録解除』
- 350 ページの『ローカル・データベースおよびノード・ディレクトリーの LDAP 項目をリフレッシュする』
- 351 ページの『LDAP ディレクトリー・パーティションまたはドメインの検索』
- 352 ページの『LDAP でのホスト・データベースの登録』
- 353 ページの『LDAP 環境でのユーザー・レベルでの DB2 レジストリー変数の設定』
- 354 ページの『インストールが完了した後に LDAP サポートを使用可能にする』
- 355 ページの『LDAP サポートを使用不可にする』
- 358 ページの『Active Directory 用にディレクトリー・スキーマを拡張する』

関連資料:

- 338 ページの『サポートされている LDAP クライアント構成およびサーバー構成』
- 360 ページの『Active Directory 内の DB2 オブジェクト』
- 368 ページの『DB2 で使用する LDAP オブジェクト・クラスおよび属性』
- 361 ページの『Netscape LDAP ディレクトリー・サポートおよび属性定義』

サポートされている LDAP クライアント構成およびサーバー構成

次の表では、サポートされる LDAP クライアント構成と LDAP サーバー構成を要約しています。

表 22. サポートされている *LDAP* クライアント構成およびサーバー構成

	IBM SecureWay Directory	Microsoft Active Directory	Netscape LDAP サーバー
IBM LDAP クライアント	サポートされる	サポートされる	サポートされる
Microsoft LDAP/ADSI クライアント	サポートされる	サポートされる	サポートされる

IBM SecureWay Directory バージョン 3.1 は、 Windows NT、AIX、 Solaris オペレーティング環境、および HP-UX で使用可能な LDAP バージョン 3 サーバーです。 SecureWay Directory は、 AIX および iSeries (AS/400) での基本オペレーティング・システムの一部として、また OS/390 Security Server と共に出荷されます。

32 ビット・バージョンの DB2 Universal Database™ (DB2 UDB) は、 IBM LDAP クライアントを AIX、Solaris オペレーティング環境、 HP-UX 11.11、 Windows、 Linux IA32、および Linux/390 でサポートしています。

Microsoft Active Directory は LDAP バージョン 3 サーバーです。 Windows 2000 Server オペレーティング・システムの一部として使用できます。

Microsoft LDAP クライアントは、Windows オペレーティング・システムに組み込まれています。

Windows オペレーティング・システムで実行されている場合、 DB2 は IBM SecureWay Directory Server へのアクセス手段として、 IBM LDAP クライアントまたは Microsoft LDAP クライアントの使用をサポートします。 IBM LDAP クライアントを明示的に選択するには、 **db2set** コマンドを使用して DB2LDAP_CLIENT_PROVIDER レジストリー変数を ″IBM″ に設定します。

関連概念:

- 337 ページの『Lightweight Directory Access Protocol (LDAP) の紹介』
- 339 ページの『Active Directory のサポート』

Active Directory のサポート

DB2® Universal Database (DB2 UDB) は、次のようにして Active Directory を活用します。

1. DB2 データベース・サーバーは、 ibm_db2Node オブジェクトとして Active Directory に公開されます。 ibm_db2Node オブジェクト・クラスは、 ServiceConnectionPoint (SCP) オブジェクト・クラスのサブクラスです。各 ibm_db2Node オブジェクトにはプロトコル構成情報が含まれていて、クライアント・アプリケーションが DB2 UDB データベース・サーバーへ接続できるようになっています。新しいデータベースが作成されると、そのデータベースは ibm_db2Node の下にある ibm_db2Database オブジェクトとして、 Active Directory に公開されます。
2. リモート・データベースに接続するときは、 DB2 クライアントは LDAP インターフェースを使用して、 Active Directory に対して ibm_db2Database オブジェクトについての照会を行います。データベース・サーバーへ接続するためのプロ

トコル通信 (バインド情報) は、 ibm_db2Database オブジェクトが作成された ibm_db2Node オブジェクトから取得されます。

ibm_db2Node および ibm_db2Database オブジェクトのプロパティー・ページは、ドメイン・コントローラーの *Active Directory Users and Computer* 管理コンソール (MMC) を使用して、表示または変更が可能です。プロパティー・ページをセットアップするには、以下のように regsrv32 コマンドを実行して、 DB2 オブジェクトのためのプロパティー・ページを登録します。

```
regsvr32 %DB2PATH%¥bin¥db2ads.dll
```

オブジェクトは、ドメイン・コントローラーの *Active Directory Users and Computer* 管理コンソール (MMC) を使用して表示することができます。この管理ツールは、「スタート」—>「プログラム」—>「管理ツール (Administration Tools)」—>「Active Directory Users and Computer」にあります。

注: コンピューター・オブジェクトの下の DB2 UDB オブジェクトを表示するには、「表示 (View)」メニューから *Users, Groups, and Computers as containers* を選択する必要があります。

注: DB2 UDB がドメイン・コントローラーにインストールされていない場合でも、ドメイン・コントローラーのローカル・ディレクトリーに、%DB2PATH%¥bin から db2ads.dll ファイルを、また %DB2PATH%¥msg¥locale-name からリソース DLL db2adsr.dll をコピーすることで、 DB2 UDB オブジェクトのプロパティー・ページを表示できます。 (コピーされたこの 2 つのファイルの保管場所は、 PATH ユーザー/システム環境変数に含まれるいずれかのディレクトリーでなければなりません。) その後、regsvr32 コマンドをローカル・ディレクトリーから実行し、 DLL を登録します。

関連概念:

- 357 ページの『Active Directory のセキュリティーに関する考慮事項』

関連タスク:

- 340 ページの『Active Directory を使用するよう DB2 を構成する』
- 358 ページの『Active Directory 用にディレクトリー・スキーマを拡張する』

関連資料:

- 360 ページの『Active Directory 内の DB2 オブジェクト』

Active Directory を使用するよう DB2 を構成する

手順:

Microsoft Active Directory へアクセスするためには、以下の条件を満たしていなければなりません。

1. DB2 Universal Database™ (DB2 UDB) を実行するマシンは、 Windows 2000 または Windows Server 2003 ドメインに属している必要がある。
2. Microsoft LDAP クライアントがインストールされている。 Microsoft® LDAP クライアントは、Windows 2000、Windows XP、および Windows Server 2003 オ

ペレーティング・システムに含まれています。 Windows 98、Windows NT、または Windows Me の場合、 Active Directory クライアント拡張機能 (wldap32.dll) がシステム・ディレクトリーに存在することを確認してください。

3. LDAP サポートが有効になっている。 Windows 2000、Windows XP、または Windows Server 2003 の場合、 LDAP サポートはインストール・プログラムによって有効になります。 Windows 98、Windows NT、または Windows Me の場合、 **db2set** コマンドを使用して DB2_ENABLE_LDAP レジストリー変数を ″YES″ に設定することにより、 LDAP を明示的に有効にしなければなりません。
4. DB2 UDB を実行して Active Directory から情報を読み取るときは、ドメイン・ユーザー・アカウントにログオンする。

関連概念:

- 339 ページの『Active Directory のサポート』
- 「*管理ガイド: パフォーマンス*」の『DB2 レジストリー変数と環境変数』

関連タスク:

- 343 ページの『DB2 アプリケーション用 LDAP ユーザーの構成』

IBM LDAP 環境での DB2 の構成

手順:

DB2 を IBM LDAP 環境で使用できるようにするには、各マシンで次のような構成を行わなければなりません。

- LDAP サポートが有効になっている。 Windows 2000 の場合、 LDAP サポートはインストール・プログラムによって有効になります。 Windows 98 または Windows NT の場合、 **db2set** コマンドを使用して DB2_ENABLE_LDAP レジストリー変数を『YES』に設定することにより、 LDAP を明示的に有効にしなければなりません。すべての Windows オペレーティング・システムは、デフォルトで Microsoft の LDAP クライアントを使用します。 IBM LDAP クライアントを使用したい場合には、 **db2set** コマンドを使用して、DB2LDAP_CLIENT_PROVIDER レジストリー変数を 『IBM』 に設定する必要があります。
- LDAP サーバーの TCP/IP ホスト名とポート番号を構成する。これらの値は DB2LDAPHOST 応答キーワードを使用して自動インストール時に入力することもできますし、 DB2SET コマンドを使用して後から手動設定することもできます。

  ```
  db2set DB2LDAPHOST=<hostname[:port]>
  ```

  ここで、hostname は LDAP サーバーの TCP/IP ホスト名、 [:port] はポート番号です。ポート番号が指定されなかった場合、 DB2 はデフォルトの LDAP ポート (389) を使用します。

  DB2 オブジェクトの探索は、LDAP 基本識別名 (baseDN) によって行われます。IBM SecureWay LDAP ディレクトリー・サーバー・バージョン 3.1 を使用している場合は、基本識別名を構成する必要はありません。 DB2 がこの情報をサーバーから動的に取得するからです。ただし、IBM eNetwork Directory Server バー

ジョン 2.1 を使用している場合は、 DB2SET コマンドを使用することにより、各マシンで LDAP 基本識別名を構成しなければなりません。

```
db2set DB2LDAP_BASEDN=<baseDN>
```

ここで、baseDN は LDAP サーバーに定義されている LDAP 接尾部の名前です。この LDAP 接尾部は DB2 オブジェクトを収容するのに使用されます。

- LDAP ユーザーの識別名 (DN) とパスワードを構成する必要があります。ただし、これらが必要になるのは、 DB2 ユーザー固有情報を保管するのに LDAP を使用する計画がある場合だけです。

関連概念:

- 「*管理ガイド: パフォーマンス*」の『DB2 レジストリー変数と環境変数』

関連タスク:

- 340 ページの『Active Directory を使用するよう DB2 を構成する』
- 342 ページの『LDAP ユーザーの作成』

関連資料:

- 「コマンド・リファレンス」の『db2set - DB2 プロファイル・レジストリー・コマンド』

LDAP ユーザーの作成

手順:

DB2 では、 DB2 レジストリー変数と CLI 構成をユーザー・レベルで設定できます。 (これは AIX および Solaris プラットフォームには当てはまりません。) ユーザー・レベルのサポートがあれば、マルチユーザー環境でユーザー固有の設定が可能です。 Windows NT Terminal Server はその一例です。ここでは、各ログオン・ユーザーが、システム環境や他のユーザーの環境を干渉することなく自分の環境をカスタマイズできます。

IBM LDAP ディレクトリーを使用する場合、ユーザー・レベルの情報を LDAP に保管する前に、 LDAP ユーザーを定義しなければなりません。 LDAP ユーザーを作成するには、以下のいずれかを実行します。

- ユーザー・オブジェクトのすべての属性を収容する LDIF ファイルを作成してから、 LDIF インポート・ユーティリティーを実行し、そのオブジェクトを LDAP ディレクトリーにインポートします。 IBM LDAP サーバー用の LDIF ユーティリティーは ″LDIF2DB″ です。
- ディレクトリー管理ツール (DMT) を使用して、ユーザー・オブジェクトを作成します。このツールは、IBM SecureWay LDAP Directory Server バージョン 3.1 でのみ使用できます。

人物オブジェクトの属性を収めた LDIF ファイルは次のようなものです。

```
File name: newuser.ldif

dn: cn=Mary Burnnet, ou=DB2 UDB Development, ou=Toronto, o=ibm, c=ca
objectclass: ePerson
cn: Mary Burnnet
sn: Burnnet
uid: mburnnet
```

```
userPassword: password
telephonenumber: 1-416-123-4567
facsimiletelephonenumber: 1-416-123-4568
title: Software Developer
```

IBM LDIF インポート・ユーティリティーを使用して LDIF ファイルをインポートする LDIF コマンドの例を、以下に示します。

```
LDIF2DB -i newuser.ldif
```

注:

1. LDIF2DB コマンドは LDAP サーバー・マシンから実行しなければなりません。
2. 必要なアクセス・コントロール・リスト (ACL) を LDAP ユーザー・オブジェクトに付与することにより、 LDAP ユーザーが自分のオブジェクトの追加、削除、読み取り、書き込みを行えるようにしなければなりません。ユーザー・オブジェクトについての ACL を付与するには、 LDAP Directory Server Web Administration ツールを使用します。

関連タスク:

- 341 ページの『IBM LDAP 環境での DB2 の構成』
- 343 ページの『DB2 アプリケーション用 LDAP ユーザーの構成』

DB2 アプリケーション用 LDAP ユーザーの構成

手順:

Microsoft LDAP クライアントを使用する場合、 LDAP ユーザーはオペレーティング・システムのユーザー・アカウントと同一です。しかし、 IBM LDAP クライアントで作業している場合、DB2 を使用する前には、現行ログオン・ユーザーの LDAP ユーザー識別名 (DN) とパスワードを構成しなければなりません。これを行うには、db2ldcfg ユーティリティーを次のように使用します。

```
db2ldcfg -u <userDN> -w <password> -> set the user's DN and password
         -r                        -> clear the user's DN and password
```

たとえば、以下のとおりです。

```
db2ldcfg -u "cn=Mary Burnnet,ou=DB2 UDB Development,ou=Toronto,o=ibm,c=ca"
         -w password
```

関連タスク:

- 342 ページの『LDAP ユーザーの作成』

関連資料:

- 「コマンド・リファレンス」の『db2ldcfg - LDAP 環境の構成コマンド』

インストール後の DB2 サーバーの登録

手順:

DB2 サーバー・インスタンスに接続する際に、クライアント・アプリケーションで使うプロトコル構成情報を公表するためには、各 DB2 サーバー・インスタンスを LDAP に登録しておく必要があります。データベース・サーバーのインスタンスを

登録するときには、*ノード名* を指定する必要があります。このノード名は、クライアント・アプリケーションがサーバーに接続するときに、そのクライアント・アプリケーションによって使われます。 **CATALOG LDAP NODE** コマンドを使用して、もう一つ、 LDAP ノードの別名をカタログすることができます。

注: Windows 2000 ドメイン環境で作業している場合は、インストール時に DB2 サーバー・インスタンスが、次の情報とともに、自動的に Active Directory へ登録されます。

```
nodename: TCP/IP hostname
protocol type: TCP/IP
```

TCP/IP ホスト名が 9 文字以上の場合は、8 文字に切り捨てられます。

REGISTER コマンドは次のようになります。

```
db2 register db2 server in ldap
   as <ldap_node_name>
   protocol tcpip
```

protocol 文節には、このデータベース・サーバーに接続するときに使用する通信プロトコルを指定します。

複数の物理マシンを含む DB2 Universal Database Enterprise Server Edition のインスタンスを作成する場合、それぞれのマシンで一度 **REGISTER** コマンドを呼び出す必要があります。 **rah** コマンドは、すべてのマシンで **REGISTER** コマンドを発行する際に使用します。

注: LDAP では名前はユニークでなければならないので、各マシンに同じ ldap_node_name を使うことはできません。 **REGISTER** コマンドでは、各マシンのホスト名を、 ldap_node_name に置き換えることができます。たとえば、以下のとおりです。

```
rah ">DB2 REGISTER DB2 SERVER IN LDAP AS <> PROTOCOL TCPIP"
```

″<>″ には、**rah** コマンドを実行している各マシンのホスト名が入ります。めったにありませんが、複数の DB2 Universal Database Enterprise Server Edition インスタンスが存在する場合、 **rah** コマンドでは、インスタンスとホスト索引の組み合わせをノード名として使用することができます。

REGISTER コマンドは、リモート DB2 サーバーに対して発行できます。そのためには、リモート・サーバーの登録時に、リモート・コンピューター名、インスタンス名、およびプロトコル構成パラメーターを指定しなければなりません。このコマンドは、次のように使います。

```
db2 register db2 server in ldap
   as <ldap_node_name>
   protocol tcpip
   hostname <host_name>
   svcename <tcpip_service_name>
   remote <remote_computer_name>
   instance <instance_name>
```

コンピューター名には、次の規則が使われます。

- TCP/IP を構成する場合、コンピューター名は TCP/IP ホスト名と同じでなければならない。

- APPN を構成する場合、パートナー LU 名をコンピューター名として使用する。

高可用性の環境あるいはフェイルオーバーの環境で実行し、通信プロトコルとして TCP/IP を使用する場合、クラスター の IP アドレスを使わなければなりません。クラスターの IP アドレスを使用するなら、クライアントは、マシンごとに個別の TCP/IP ノードをカタログすることなく、いずれかのマシンでサーバーに接続することができます。 このクラスターの IP アドレスは、次に示すように、 hostname 文節を使って指定します。

```
db2 register db2 server in ldap
   as <ldap_node_name>
   protocol tcpip
   hostname n.nn.nn.nn
```

n.nn.nn.nn はクラスターの IP アドレスです。

関連概念:

- 379 ページの『rah および db2_all コマンドの概要』

関連タスク:

- 345 ページの『DB2 サーバー用のプロトコル情報を更新する』
- 347 ページの『アタッチのためにノード別名をカタログする』
- 347 ページの『DB2 サーバーの登録を解除する』
- 348 ページの『LDAP 環境でのリモート・サーバーのアタッチ』

関連資料:

- 「コマンド・リファレンス」の『REGISTER コマンド』
- 「コマンド・リファレンス」の『CATALOG LDAP NODE コマンド』

DB2 サーバー用のプロトコル情報を更新する

手順:

LDAP 内の DB2 サーバー情報は、最新のものにしておく必要があります。たとえば、プロトコル構成パラメーターまたはサーバーのネットワーク・アドレスを変更するときには、 LDAP を更新しなければなりません。

ローカル・マシンの LDAP にある DB2 サーバーを更新するには、次のコマンドを使用します。

```
db2 update ldap ...
```

更新可能なプロトコル構成パラメーターの例としては、以下のものがあります。

- TCP/IP ホスト名およびサービス名パラメーターまたはポート番号パラメーター
- TP 名、パートナー LU、あるいはモードのような、APPC プロトコル情報
- NetBIOS ワークステーション名

リモート DB2 サーバーのプロトコル構成パラメーターを更新するには、 node 文節を指定した UPDATE LDAP コマンドを使います。

```
db2 update ldap
   node <node_name>
   hostname <host_name>
   svcename <tcpip_service_name>
```

関連タスク:

- 343 ページの『インストール後の DB2 サーバーの登録』
- 347 ページの『アタッチのためにノード別名をカタログする』
- 348 ページの『LDAP 環境でのリモート・サーバーのアタッチ』

関連資料:

- 「コマンド・リファレンス」の『UPDATE LDAP NODE コマンド』

他のサーバーへの LDAP クライアントの転送

システム障害時のクライアント転送と同じような機能が、 LDAP でも使用できます。

前提条件:

DB2_ENABLE_LDAP レジストリー変数を「Yes」に設定します。

手順:

LDAP 環境では、データベースおよびノード・ディレクトリーの情報がすべて LDAP サーバーで維持されます。クライアントは、LDAP ディレクトリーから情報を検索します。 DB2LDAPCACHE レジストリー変数が「Yes」に設定されている場合、この情報はローカル・データベースおよびノード・ディレクトリーで更新されます。

UPDATE ALTERNATE SERVER FOR LDAP DATABASE コマンドを使用して、LDAP で DB2 Universal Database™ (DB2 UDB) を表すデータベースの ALTERNATE SERVER を定義します。

この ALTERNATE SERVER 情報が確立されると、接続時にこの情報がクライアントに戻されます。

注: UPDATE ALTERNATE SERVER FOR DATABASE コマンド (「FOR LDAP DATABASE」ではないことに注意) をサーバー・インスタンスで入力すると、サーバーで LDAP サポートが使用可能 (DB2_ENABLE_LDAP=Yes) になっていて、かつ (**db2ldcfg** が実行済みで) LDAP ユーザー ID とパスワードがすでにキャッシュされている場合、データベースの ALTERNATE SERVER が LDAP サーバー上で自動的に (つまり暗黙的に) 更新されます。これは、UPDATE ALTERNATE SERVER FOR LDAP DATABASE を明示的に入力したかのように機能します。

関連概念:

- 81 ページの『自動クライアント・リルートのインプリメンテーション』
- 331 ページの『クライアントの自動転送についての説明およびセットアップ』

アタッチのためにノード別名をカタログする

手順:

DB2 サーバーのノード名については、サーバーを LDAP に登録するときに、指定しなければなりません。アプリケーションはこのノード名を使用して、データベース・サーバーにアタッチします。ノード名がアプリケーション内でハード・コーディングされる場合のように、別のノード名が必要であれば、CATALOG LDAP NODE コマンドを使用して変更してください。 このコマンドは、次のようになります。

```
db2 catalog ldap node <ldap_node_name>
   as <new_alias_name>
```

LDAP ノードをアンカタログするには、UNCATALOG LDAP NODE コマンドを使用します。このコマンドは、次のようになります。

```
db2 uncatalog ldap node <ldap_node_name>
```

関連タスク:

- 343 ページの『インストール後の DB2 サーバーの登録』
- 348 ページの『LDAP 環境でのリモート・サーバーのアタッチ』

関連資料:

- 「コマンド・リファレンス」の『CATALOG LDAP NODE コマンド』
- 「コマンド・リファレンス」の『UNCATALOG LDAP NODE コマンド』

DB2 サーバーの登録を解除する

手順:

LDAP から特定のインスタンスの登録を解除すると、そのインスタンスを参照するすべてのノード、あるいは、そのインスタンスを参照する別名、オブジェクト、およびデータベース・オブジェクトも除去されます。

ローカルまたはリモート・マシンのいずれでも、DB2 サーバーの登録を解除する場合には、そのサーバーの LDAP ノード名を指定する必要があります。

```
db2 deregister db2 server in ldap
   node <node_name>
```

DB2 サーバーの登録を解除すると、 DB2 サーバーのその同じインスタンスを参照する LDAP ノード項目および LDAP データベース項目も、すべてアンカタログされます。

関連タスク:

- 343 ページの『インストール後の DB2 サーバーの登録』

関連資料:

- 「コマンド・リファレンス」の『DEREGISTER コマンド』

LDAP ディレクトリーへのデータベースの登録

手順:

インスタンス内でデータベースを作成するときに、そのデータベースは LDAP へ自動的に登録されます。登録することにより、クライアント・マシンでデータベースおよびノードをカタログせずに、データベースへリモート・クライアント接続できるようになります。 クライアントがデータベースへ接続しようとする場合に、ローカル・マシンのデータベース・ディレクトリーにそのデータベースが存在していなければ、 LDAP ディレクトリーが検索されます。

LDAP ディレクトリーにその名前が存在する場合、データベースはローカル・マシンで作成されますが、 LDAP ディレクトリーにおける名前の競合を警告する警告メッセージが戻されます。このため、LDAP ディレクトリーでは、手動でデータベースをカタログすることができます。ユーザーは、CATALOG LDAP DATABASE コマンドを使用して、リモート・サーバーのデータベースを LDAP に登録できます。リモート・データベースを登録するときには、リモート・データベース・サーバーを表す LDAP ノードの名前を指定します。リモート・データベース・サーバーは、データベースを登録する**前に**、 REGISTER DB2 SERVER IN LDAP コマンドを使用して、 LDAP に登録**しなければなりません**。

データベースを手動で LDAP に登録するには、CATALOG LDAP DATABASE コマンドを使います。

```
db2 catalog ldap database <dbname>
   at node <node_name>
   with "My LDAP database"
```

関連タスク:

- 343 ページの『インストール後の DB2 サーバーの登録』
- 349 ページの『LDAP ディレクトリーからのデータベースの登録解除』

関連資料:

- 「コマンド・リファレンス」の『CATALOG LDAP DATABASE コマンド』

LDAP 環境でのリモート・サーバーのアタッチ

手順:

LDAP 環境では、ATTACH コマンドで LDAP ノード名を使用することにより、リモート・データベース・サーバーへアタッチすることができます。

```
db2 attach to <ldap_node_name>
```

クライアント・アプリケーションが特定のノードまたはデータベースに初めて接続する場合、そのノードはローカル・ノード・ディレクトリーに含まれていないので、データベース・マネージャーは LDAP ディレクトリーの中でその宛先ノード項目を探します。 LDAP ディレクトリーでその項目が見つかったら、リモート・サーバーのプロトコル情報が取り出されます。データベースに接続し、LDAP ディレクトリーでその項目が見つかったら、データベース情報が取り出されます。この情報を使用して、データベース・マネージャーはローカル・マシンのデータベース項目

およびノード項目を自動的にカタログします。次回にクライアント・アプリケーションが同じノードまたはデータベースにアタッチするときには、ローカル・データベース・ディレクトリーの情報が使われるので、 LDAP ディレクトリーを検索する必要はありません。

詳細情報: キャッシュ・メカニズムが備えられているので、クライアントは LDAP サーバーを一度だけ検索します。いったん情報を検索したら、 *dir_cache* データベース・マネージャー構成パラメーターおよび DB2LDAPCACHE レジストリー変数の値に基づいて、その情報はローカル・マシンに格納されるか、キャッシュに入れられます。

- DB2LDAPCACHE=NO かつ *dir_cache*=NO の場合、情報は必ず LDAP から読み取られます。
- DB2LDAPCACHE=NO かつ *dir_cache*=YES の場合、 LDAP から一度情報が読み取られ、その情報は DB2 キャッシュに挿入されます。
- DB2LDAPCACHE=YES であるか、これが設定されない場合には、 LDAP サーバーから情報が一度読み取られ、その情報はローカル・データベース、ノード、および DCS ディレクトリーにキャッシュされます。

注: LDAP 情報のキャッシュには、ユーザー・レベルの CLI や DB2 プロファイル・レジストリー変数は含まれません。

関連概念:

- 「*管理ガイド: パフォーマンス*」の『DB2 レジストリー変数と環境変数』

関連タスク:

- 343 ページの『インストール後の DB2 サーバーの登録』
- 345 ページの『DB2 サーバー用のプロトコル情報を更新する』
- 347 ページの『アタッチのためにノード別名をカタログする』
- 348 ページの『LDAP ディレクトリーへのデータベースの登録』

関連資料:

- 「コマンド・リファレンス」の『ATTACH コマンド』

LDAP ディレクトリーからのデータベースの登録解除

手順:

以下の場合に、データベースは LDAP から自動的に登録解除されます。

- データベースをドロップした場合。
- 所有インスタンスが LDAP から登録解除された場合。

データベースは、次のようにして、LDAP から手動で登録解除することができます。

```
db2 uncatalog ldap database <dbname>
```

関連タスク:

- 348 ページの『LDAP ディレクトリーへのデータベースの登録』

関連資料:

- 「コマンド・リファレンス」の『UNCATALOG LDAP DATABASE コマンド』

ローカル・データベースおよびノード・ディレクトリーの LDAP 項目をリフレッシュする

手順:

LDAP 情報は変更されることがあるため、ローカルおよびノード・ディレクトリーの LDAP 項目をリフレッシュする必要があります。 LDAP 内の項目をキャッシュするには、ローカル・データベースおよびノード・ディレクトリーが使われます。

詳細情報: キャッシュ・メカニズムが備えられているので、クライアントは LDAP サーバーを一度だけ検索します。いったん情報を検索したら、 *dir_cache* データベース・マネージャー構成パラメーターおよび DB2LDAPCACHE レジストリー変数の値に基づいて、その情報はローカル・マシンに格納されるか、キャッシュに入れられます。

- DB2LDAPCACHE=NO かつ *dir_cache*=NO の場合、情報は必ず LDAP から読み取られます。
- DB2LDAPCACHE=NO かつ *dir_cache*=YES の場合、 LDAP から一度情報が読み取られ、その情報は DB2 キャッシュに挿入されます。
- DB2LDAPCACHE=YES であるか、これが設定されない場合には、 LDAP サーバーから情報が一度読み取られ、その情報はローカル・データベース、ノード、および DCS ディレクトリーにキャッシュされます。

注: LDAP 情報のキャッシュには、ユーザー・レベルの CLI や DB2 プロファイル・レジストリー変数は含まれません。

LDAP リソースを参照するデータベース項目をリフレッシュするには、次のコマンドを使用します。

```
db2 refresh ldap database directory
```

LDAP リソースを参照するローカル・マシンのノード項目をリフレッシュするには、次のコマンドを使います。

```
db2 refresh ldap node directory
```

リフレッシュの一環として、ローカル・データベースおよびノード・ディレクトリーに保管されているすべての LDAP 項目が除去されます。次回にアプリケーションがデータベースまたはノードにアクセスすると、情報を LDAP から直接に読み取り、ローカル・データベースまたはノード・ディレクトリーに新しい項目を生成します。

リフレッシュを周期的に実行するには、以下を行います。

- 周期的に実行するリフレッシュをスケジュールする。
- システムのブート時に REFRESH コマンドを実行する。
- 使用可能な管理パッケージを使い、すべてのクライアント・マシンで REFRESH コマンドを呼び出す。
- DB2LDAPCACHE=″NO″ を設定して、 LDAP 情報がデータベース、ノード、および DCS ディレクトリーのキャッシュに入れられないようにする。

関連概念:

- 「*管理ガイド: パフォーマンス*」の『DB2 レジストリー変数と環境変数』

関連資料:

- 「*管理ガイド: パフォーマンス*」の『dir_cache - 「ディレクトリー・キャッシュ・サポート」構成パラメーター』
- 「コマンド・リファレンス」の『REFRESH LDAP コマンド』

LDAP ディレクトリー・パーティションまたはドメインの検索

手順:

DB2 UDB は、現行の LDAP ディレクトリー・パーティションを検索します (Windows 2000 環境の場合は、現行の Active Directory ドメインを検索します)。複数の LDAP ディレクトリー・パーティションまたはドメインがある環境では、検索の範囲を設定することができます。たとえば、現在のパーティションまたはドメインに情報がなければ、他のパーティションまたはドメインすべての自動検索を要求できます。一方、検索の範囲については、ローカル・マシンのみの検索に制限することも可能です。

この検索範囲は、DB2 UDB プロファイルのレジストリー変数 DB2LDAP_SEARCH_SCOPE によって制御されます。 LDAP において、検索範囲の値をグローバル・レベルで設定するには、 *db2set* コマンドで ″-gl″ オプションを使用します。これは「LDAP 内でグローバル」という意味です。

```
db2set -gl db2ldap_search_scope=<value>
```

可能な値には、″local″、″domain″、あるいは ″global″ があります。デフォルト値は ″domain″ です。この値は、検索範囲を現行のディレクトリー・パーティションに限定します。 LDAP での検索範囲を設定すると、全社的なデフォルトの検索範囲を設定できるようになります。たとえば、新しいデータベースを作成したら、検索範囲を ″global″ に初期設定することができます。このように設定すれば、どのクライアント・マシンであっても、他のすべてのパーティションまたはドメインを検索して、その中から特定のパーティションまたはドメインで定義されているデータベースを見つけられます。最初にそれぞれのクライアントに接続してから、各マシンに項目を記録したら、検索範囲を ″local″ に変更することができます。 ″local″ に変更すると、各クライアントはどのパーティションまたはドメインもスキャンしません。

注: LDAP 内のグローバル・レベルでの変数の設定をサポートするレジストリー変数は、 DB2 UDB プロファイルのレジストリー変数 DB2LDAP_KEEP_CONNECTION および DB2LDAP_SEARCH_SCOPE のみです。

関連概念:

- 「*管理ガイド: パフォーマンス*」の『DB2 レジストリー変数と環境変数』

関連タスク:

- 32 ページの『レジストリーおよび環境変数の宣言』

LDAP でのホスト・データベースの登録

手順:

ホスト・データベースを LDAP で登録するときは、2 通りの構成が可能です。

- ホスト・データベースへの直接接続、または
- ホスト・データベースへのゲートウェイ経由の接続

前者の場合、ユーザーはホスト・サーバーを LDAP に登録してから、そのホスト・サーバーのノード名を指定してホスト・データベースを LDAP でカタログします。後者の場合、ユーザーはゲートウェイ・サーバーを LDAP に登録してから、そのゲートウェイ・サーバーのノード名を指定して LDAP でホスト・データベースをカタログします。

LDAP サポートが DB2® コネクト・ゲートウェイにおいて有効で、データベースがゲートウェイ・データベース・ディレクトリーでは見つからなかった場合、 DB2 UDB は LDAP を検索し、見つかった情報を保持しようとします。

上記の 2 つの場合の例として、次のような状況を考慮してみましょう。NIAGARA_FALLS という名前のホスト・データベースがあるとします。このホスト・データベースは、APPN と TCP/IP を使用して着信接続を受け入れることができます。あるクライアントが DB2 Connect をインストールしていないためにそのホストへ直接接続できない場合は、 ″goto@niagara″ という名前のゲートウェイを使用してホストに接続します。

以下のステップを実行する必要があります。

1. APPN 接続を可能にするため、ホスト・データベース・サーバーを LDAP に登録します。 REMOTE 文節と INSTANCE 文節は任意です。 NODETYPE 文節を ″DCS″ に設定して、これがホスト・データベース・サーバーであることを示します。

   ```
   db2 register ldap as nfappn appn network CAIBMOML partnerlu NFLU
      mode IBMRDB remote mvssys instance msvinst nodetype dcs
   ```

2. TCP/IP 接続を可能にするため、ホスト・データベース・サーバーを LDAP に登録します。このサーバーの TCP/IP ホスト名は ″myhost″、ポート番号は ″446″ です。ステップ 1 と同じように、NODETYPE 文節を ″DCS″ に設定して、これがホスト・データベース・サーバーであることを示します。

   ```
   db2 register ldap as nftcpip tcpip hostname myhost svcename 446
      remote mvssys instance mvsinst nodetype dcs
   ```

3. TCP/IP 接続を可能にするため、DB2 Connect ゲートウェイ・サーバーを LDAP に登録します。このゲートウェイ・サーバーの TCP/IP ホスト名は ″niagara″、ポート番号は ″50000″ です。

   ```
   db2 register ldap as whasf tcpip hostname niagara svcename 50000
      remote niagara instance goto nodetype server
   ```

4. TCP/IP 接続を使用してホスト・データベースを LDAP でカタログします。このホストのデータベース名は ″NIAGARA_FALLS″、データベース別名は ″nftcpip″ です。 GWNODE 文節は、DB2 Connect ゲートウェイ・サーバーのノード名を指定するために使用されています。

```
db2 catalog ldap database NIAGARA_FALLS as nftcpip at node nftcpip
   gwnode whasf authentication dcs
```

5. APPN 接続を使用してホスト・データベースを LDAP でカタログします。

```
db2 catalog ldap database NIAGARA_FALLS as nfappn at node nfappn
   gwnode whasf authentication dcs
```

上記のように登録とカタログが完了した後、 TCP/IP を使用してホストに接続したい場合は、″nftcpip″ に接続します。 APPN を使用してホストに接続したい場合は、″nfappn″ に接続します。 DB2 Connect がクライアント・ワークステーションにインストールされていない場合、接続は TCP/IP によってゲートウェイ経由で確立されます。そこからホストまでの部分については、″nftcpip″ を使用する場合は TCP/IP によって、 ″nfappn″ を使用する場合は APPN によって接続されます。

通常は、LDAP においてホスト・データベース情報を手動で構成できるので、それぞれのクライアントでは、各マシンのデータベースおよびノードをローカルに手動でカタログする必要はありません。次のように処理します。

1. ホスト・データベース・サーバーを LDAP に登録します。そのためには、ホスト・データベース・サーバーのリモート・コンピューター名、インスタンス名、およびノード・タイプを、REGISTER コマンドの REMOTE 文節、INSTANCE 文節、および NODETYPE 文節にそれぞれ指定しなければなりません。
   REMOTE 文節は、ホスト・サーバー・マシンのホスト名と LU 名のどちらに設定しても構いません。 INSTANCE 文節は、8 文字以下の文字ストリングで設定します。 (たとえば、インスタンス名を ″DB2″ のように設定します。) NODE TYPE 文節は必ず ″DCS″ に設定して、これがホスト・データベース・サーバーであることを示さなければなりません。
2. CATALOG LDAP DATABASE コマンドを使ってホスト・データベースを LDAP に登録します。さらに、PARMS 文節を使用して、任意の DRDA パラメーターを指定することができます。データベース認証タイプは ″DCS″ に設定してください。

関連資料:

- 「コマンド・リファレンス」の『REGISTER コマンド』
- 「コマンド・リファレンス」の『CATALOG LDAP DATABASE コマンド』

LDAP 環境でのユーザー・レベルでの DB2 レジストリー変数の設定

手順:

LDAP 環境では、DB2 プロファイルのレジストリー変数をユーザー・レベルで設定できます。これにより、それぞれの DB2 環境をカスタマイズすることができます。 DB2 プロファイルのレジストリー変数をユーザー・レベルで設定するには、`-ul` オプションを使います。

```
db2set -ul <variable>=<value>
```

注: これは AIX や Solaris オペレーティング環境ではサポートされません。

DB2 はキャッシュ・メカニズムを備えています。 DB2 プロファイルのユーザー・レベルのレジストリー変数は、ローカル・マシンにキャッシュされます。 `-ul` パラ

メーターを指定すると、DB2 は、いつもキャッシュから DB2 レジストリー変数を読み取ります。キャッシュは、以下の場合に更新されます。

- DB2 レジストリー変数をユーザー・レベルで更新またはリセットした場合。
- LDAP プロファイル変数をユーザー・レベルで更新するためのコマンドは、次のとおりです。

```
db2set -ur
```

関連タスク:

- 32 ページの『レジストリーおよび環境変数の宣言』

関連資料:

- 「コマンド・リファレンス」の『db2set - DB2 プロファイル・レジストリー・コマンド』

インストールが完了した後に LDAP サポートを使用可能にする

手順:

インストールの作業が完了した後のある時点で LDAP サポートを使用可能にするには、各マシンで以下のような手順に従います。

- LDAP サポートのバイナリー・ファイルをインストールします。セットアップ・プログラムを実行し、**カスタム・インストール**から、 LDAP Directory Exploitation サポートを選択します。セットアップ・プログラムによりバイナリー・ファイルがインストールされ、 DB2 プロファイルのレジストリー変数 DB2_ENABLE_LDAP が ″YES″ に設定されます。

  注: Windows 98、Windows NT、および UNIX プラットフォームの場合、 **db2set** コマンドを使用して DB2_ENABLE_LDAP レジストリー変数を ″YES″ に設定することにより、 LDAP を明示的に有効にしなければなりません。

- (UNIX プラットフォームの場合のみ) 以下のコマンドを使用して、 LDAP サーバーの TCP/IP ホスト名と (オプショナル) ポート番号を宣言します。

  ```
  db2set DB2LDAPHOST=<base_domain_name>[:port_number]
  ```

  ここで、base_domain_name は LDAP サーバーの TCP/IP ホスト名、 [:port] はポート番号です。ポート番号が指定されなかった場合、 DB2 はデフォルトの LDAP ポート (389) を使用します。

  DB2 オブジェクトの探索は、LDAP 基本識別名 (baseDN) によって行われます。IBM SecureWay LDAP ディレクトリー・サーバー・バージョン 3.1 を使用している場合は、基本識別名を構成する必要はありません。 DB2 がこの情報をサーバーから動的に取得するからです。ただし、IBM eNetwork Directory Server バージョン 2.1 を使用している場合は、 DB2SET コマンドを使用することにより、各マシンで LDAP 基本識別名を構成しなければなりません。

  ```
  db2set DB2LDAP_BASEDN=<baseDN>
  ```

  ここで、baseDN は LDAP サーバーに定義されている LDAP 接尾部の名前です。この LDAP 接尾部は DB2 オブジェクトを収容するのに使用されます。

- REGISTER LDAP AS コマンドを使用し、 DB2 サーバーの現在のインスタンスを LDAP に登録します。たとえば、以下のとおりです。

```
db2 register ldap as <node-name> protocol tcpip
```

- LDAP に登録したいデータベースがあれば、CATALOG LDAP DATABASE コマンドを実行します。たとえば、以下のとおりです。

```
db2 catalog ldap database <dbname> as <alias_dbname>
```

- LDAP ユーザーの識別名 (DN) とパスワードを入力します。ただし、これらが必要になるのは、 DB2 ユーザー固有情報を保管するのに LDAP を使用する計画がある場合だけです。

関連概念:

- 「*管理ガイド: パフォーマンス*」の『DB2 レジストリー変数と環境変数』

関連タスク:

- 355 ページの『LDAP サポートを使用不可にする』

関連資料:

- 「コマンド・リファレンス」の『REGISTER コマンド』
- 「コマンド・リファレンス」の『db2set - DB2 プロファイル・レジストリー・コマンド』
- 「コマンド・リファレンス」の『CATALOG LDAP DATABASE コマンド』

LDAP サポートを使用不可にする

手順:

LDAP サポートを使用不可にするには、以下の手順に従います。

- DB2 サーバーのインスタンスごとに、DB2 サーバーを LDAP から登録解除します。

```
db2 deregister db2 server in ldap node <nodename>
```

- DB2 プロファイルのレジストリー変数 DB2_ENABLE_LDAP を ″NO″ に設定します。

関連タスク:

- 32 ページの『レジストリーおよび環境変数の宣言』
- 354 ページの『インストールが完了した後に LDAP サポートを使用可能にする』

関連資料:

- 「コマンド・リファレンス」の『DEREGISTER コマンド』

LDAP サポートと DB2 Connect

LDAP サポートが DB2® コネクト・ゲートウェイにおいて有効で、データベースがゲートウェイ・データベース・ディレクトリーでは見つからなかった場合、 DB2 は LDAP を検索し、見つかった情報を保持しようとします。

関連概念:

- 337 ページの『Lightweight Directory Access Protocol (LDAP) の紹介』

• 356 ページの『LDAP 環境でのセキュリティーに関する考慮事項』

LDAP 環境でのセキュリティーに関する考慮事項

LDAP ディレクトリーの情報にアクセスする前に、アプリケーションまたはユーザーは、LDAP サーバーによって認証されます。この認証作業のことを、LDAP サーバーに対するバインド といいます。

匿名のユーザーが LDAP ディレクトリーに格納されている情報を追加、削除、あるいは変更してしまわないように、その情報にアクセス・コントロールを設定することは重要です。

アクセス・コントロールは、デフォルトでコンテナー・レベルで継承され、適用できるものです。新しいオブジェクトが作成される場合、その新しいオブジェクトは、親オブジェクトと同じセキュリティー属性を継承します。コンテナー・オブジェクトにアクセス・コントロールを定義するには、 LDAP サーバー用の管理ツールを使用できます。

デフォルトでは、アクセス・コントロールは以下のように定義されています。

- LDAP 内のデータベースおよびノード項目については、すべて (つまり任意の匿名ユーザー) が読み取りアクセス権限を持っています。読み取り/書き込みアクセス権限を持っているのは、ディレクトリー管理者と、そのオブジェクトの所有者または作成者だけです。
- ユーザー・プロファイルについては、プロファイル所有者と、ディレクトリー管理者が読み取り/書き込みアクセス権限を持っています。他のユーザーのプロファイルにアクセスしようとしているユーザーは、ディレクトリー管理者権限を持っていなければ、そこにアクセスできません。

注: 許可検査は必ず LDAP サーバーによって実行されます。 DB2® によって実行されることはありません。 LDAP 許可検査は DB2 認証とは関係がありません。 SYSADM 権限を持つアカウントまたは許可 ID であっても、 LDAP ディレクトリーに対するアクセス権は持っていない可能性があります。

LDAP コマンドまたは API を実行するときに、バインド識別名 (bindDN) とパスワードを指定しなかった場合、 DB2 はデフォルトの証明書を使用して LDAP サーバーにバインドします。この証明書の権限では要求したコマンドを実行できない場合もあります。その場合、エラーが戻されます。

DB2 コマンドまたは API では、 USER 文節と PASSWORD 文節を使用してユーザーの bindDN とパスワードを明示的に指定できます。

関連概念:

• 357 ページの『Active Directory のセキュリティーに関する考慮事項』

Active Directory のセキュリティーに関する考慮事項

Active Directory において、DB2® データベースとノードの各オブジェクトは、DB2 サーバーがインストールされているマシンのコンピューター・オブジェクトの下に作成されます。データベース・サーバーを Active Directory に登録したり、データベースのカタログを Active Directory に作成するためには、コンピューター・オブジェクトの下でオブジェクトを作成または更新できるアクセス権限が必要です。

デフォルトでは、コンピューター・オブジェクトの下にあるオブジェクトは、認証を受けたすべてのユーザーが読み取ることができます。管理者 (Administrators、Domain Administrators、および Enterprise Administrators グループに属するユーザー) であれば更新することもできます。特定のユーザーまたはグループにアクセス権限を付与するには、 *Active Directory Users and Computer* 管理コンソール (MMC) を次のようにして使います。

1. *Active Directory Users and Computer* 管理ツールを開始します。

   (「スタート」—>「プログラム」—>「管理ツール (Administration Tools)」—>「Active Directory Users and Computer」)
2. 「*表示 (View)*」から「*詳細 (Advanced Features)*」を選択します。
3. 「*コンピューター (Computers)*」コンテナーを選択します。
4. DB2 がインストールされているサーバー・マシンを表すコンピューター・オブジェクトの上を右クリックし、「*プロパティー (Properties)*」を選択します。
5. 「*セキュリティー (Security)*」タブを選択し、指定したユーザーまたはグループに必要なアクセス権限を追加します。

ユーザー・レベルの DB2 レジストリー変数および CLI 設定は、ユーザー・オブジェクトの下にある DB2 プロパティー・オブジェクトで保守されます。 DB2 レジストリー変数または CLI 設定をユーザー・レベルで設定するためには、そのユーザーに「ユーザー (User)」オブジェクトの下にオブジェクトを作成できるアクセス権限が必要です。

デフォルトでは、「ユーザー (User)」オブジェクトの下にオブジェクトを作成できるのは管理者だけです。 DB2 レジストリー変数または CLI 設定をユーザー・レベルで設定できるアクセス権限をユーザーに付与するには、 *Active Directory Users and Computer* 管理コンソール (MMC) を次のようにして使います。

1. *Active Directory Users and Computer* 管理ツールを開始します。

   (「スタート」—>「プログラム」—>「管理ツール (Administration Tools)」—>「Active Directory Users and Computer」)
2. 「ユーザー (Users)」コンテナーの下にあるユーザー・オブジェクトを選択します。
3. ユーザー・オブジェクトの上を右クリックし、「*プロパティー (Properties)*」を選択します。
4. 「*セキュリティー (Security)*」タブを選択します。
5. 「追加 (Add)」ボタンを使用して、ユーザー名をリストに追加します。

6. 「書き込み (Write)」および「すべての子オブジェクトの作成 (Create All Child Objects)」アクセスを付与します。
7. 「拡張 (Advanced)」設定を使用して、「このオブジェクトとすべての子オブジェクト (This object and all child objects)」に適用される許可を設定します。
8. 「継承可能な許可を親からこのオブジェクトへ継承するのを許可する (Allow inheritable permissions from parent to propagate to this object)」チェック・ボックスを選択します。

関連概念:

• 356 ページの『LDAP 環境でのセキュリティーに関する考慮事項』

DB2 オブジェクト・クラスおよび属性を使用して LDAP ディレクトリー・スキーマを拡張する

LDAP ディレクトリー・スキーマは、オブジェクト・クラスおよび属性を定義し、情報を LDAP ディレクトリー項目に格納します。オブジェクト・クラスは、一連の必須および任意指定属性で構成されています。 LDAP ディレクトリーの各項目には、それぞれに関連付けられたオブジェクト・クラスがあります。

DB2® が情報を LDAP へ格納する前に、LDAP サーバーのディレクトリー・スキーマには、 DB2 で使用するオブジェクト・クラスと属性が含まれていなければなりません。新しいオブジェクト・クラスおよび属性を基本スキーマに追加する作業のことを、ディレクトリー・スキーマの拡張といいます。

注: IBM® SecureWay® LDAP Directory v3.1 を使用している場合、 DB2 UDB バージョン 8.1 またはそれ以前によって必要とされるオブジェクト・クラスおよび属性はすべて、基本スキーマに含まれます。この場合、DB2 オブジェクト・クラスと属性を使って基本スキーマを拡張する必要はありません。ただし、DB2 UDB バージョン 8.2 には、基本スキーマに含まれない新しい属性が 2 つあります。この場合、2 つの新しい DB2 UDB 属性を使って基本スキーマを拡張する必要があります。

関連概念:

• 363 ページの『IBM SecureWay Directory Server 用にディレクトリー・スキーマを拡張する』

関連タスク:

• 358 ページの『Active Directory 用にディレクトリー・スキーマを拡張する』

Active Directory 用にディレクトリー・スキーマを拡張する

手順:

DB2 Universal Database™ (DB2 UDB) が Active Directory に情報を保管できるようにするためには、ディレクトリー・スキーマを拡張して、新しい DB2 UDB オブジェクト・クラスおよび属性を組み込む必要があります。新しいオブジェクト・クラスおよび属性をディレクトリー・スキーマに追加する作業のことを、スキーマの拡張 といいます。

Windows 2000 ドメインに属するマシンに DB2 UDB を初めてインストールする前に、 DB2 UDB スキーマ・インストール・プログラム **db2schex** を実行して、Active Directory のスキーマを拡張しておく必要があります。

db2schex プログラムは、本製品の CD-ROM の中にあります。このプログラムは、CD-ROM 内の db2 ディレクトリー、 windows サブディレクトリー、utilities サブディレクトリーにあります。たとえば、以下のとおりです。

```
x:¥db2¥windows¥utilities¥
```

ここで、x: は CD-ROM ドライブです。

コマンドは、次のように使用します。

```
db2schex
```

このコマンドには、他にも以下のような任意指定の文節があります。

- -b UserDN

  ユーザーの識別名を指定します。

- -w Password

  バインド・パスワードを指定します。

- -u

  スキーマをアンインストールします。

- -k

  エラーを無視して、アンインストールを強制的に続行します。

注:

1. UserDN とパスワードを指定しなかった場合、 **db2schex** は現在ログオンしているユーザーが指定されたものとしてバインドします。
2. userDN 文節には Windows NT ユーザー名を指定できます。
3. スキーマを更新するためには、Schema Administrators グループのメンバーであるか、スキーマを更新するための権限を委任されている必要があります。

ディレクトリー・スキーマを拡張するには、 DB2 UDB バージョン 8.2 製品に付属の **db2schex.exe** コマンドを実行する必要があります。

Windows 用 DB2 製品の以前のバージョンに付属していた **db2schex.exe** コマンドをすでに実行した場合、 DB2 UDB バージョン 8.2 に付属の同じコマンドを再び実行すると、以下の 2 つのオプション属性が ibm-db2Database クラスに追加されます。

```
ibm-db2AltGwPtr
ibm-db2NodePtr
```

Windows 用 DB2 UDB 製品の以前のバージョンに付属していた **db2schex.exe** コマンドをまだ実行していない場合、 DB2 バージョン 8.2 に付属の同じコマンドを実行すると、 DB2 UDB LDAP サポート用のすべてのクラスおよび属性が追加されます。

次に例を示します。

- DB2 UDB スキーマをインストールする方法:

```
db2schex
```

- DB2 UDB スキーマをインストールし、バインド DN とパスワードを指定する方法:

```
db2schex -b "cn=A Name,dc=toronto1,dc=ibm,dc=com"
         -w password
```

  または

```
db2schex -b Administrator -w password
```

- DB2 UDB スキーマをアンインストールする方法:

```
db2schex -u
```

- DB2 UDB スキーマをアンインストールし、エラーを無視する方法:

```
db2schex -u -k
```

Active Directory 用の DB2 UDB スキーマ・インストール・プログラムは、以下のタスクを実行します。

注:

1. どのサーバーが Schema Master であるかを検出します。
2. Schema Master となっているドメイン・コントローラーにバインドします。
3. スキーマにクラスや属性を追加する権限をユーザーに付与します。
4. Schema Master を書き込み可能にします (つまり、レジストリー内の安全インターロックを取り除きます)。
5. すべての新しい属性を作成します。
6. すべての新しいオブジェクト・クラスを作成します。
7. エラーがあるかどうか検出し、エラーが発生している場合には、スキーマに加えられた変更をロールバックするようプログラムに指示します。

関連概念:

- 358 ページの『DB2 オブジェクト・クラスおよび属性を使用して LDAP ディレクトリー・スキーマを拡張する』

Active Directory 内の DB2 オブジェクト

DB2 は、Active Directory 内の以下の 2 箇所にオブジェクトを作成します。

1. DB2 データベースおよびノード・オブジェクトが、 DB2 サーバーがインストールされているマシンのコンピューター・オブジェクトの下に作成されます。Windows NT ドメインに属していない DB2 サーバー・マシンの場合は、 DB2 データベースおよびノード・オブジェクトは「システム (System)」コンテナーの下に作成されます。
2. ユーザー・レベルの DB2 レジストリー変数および CLI 設定が、「ユーザー (User)」オブジェクトの下にある DB2 プロパティー・オブジェクトに保管されます。これらのオブジェクトには、そのユーザーに固有の情報が入ります。

関連資料:

• 368 ページの『DB2 で使用する LDAP オブジェクト・クラスおよび属性』

Netscape LDAP ディレクトリー・サポートおよび属性定義

Netscape LDAP Server のサポート・レベルは v4.12 以降です。

Netscape LDAP Server バージョン 4.12 以降では、 Netscape Directory Server を使用すると、アプリケーションは属性およびオブジェクト・クラス定義を 2 つのファイル slapd.user_oc.conf および slapd.user_at.conf に追加することによって、スキーマを拡張できます。これらの 2 つのファイルは、

```
<Netscape_install path>¥slapd-<machine_name>¥config
```

ディレクトリーにあります。

注: iPlan Directory Server 5.0 を使用する場合は、製品に付属の資料を調べて、スキーマの拡張方法についての詳細を確認する必要があります。

以下のようにして、 DB2 属性を slapd.user_at.conf に追加する必要があります。

```
########################################################################
#
# IBM DB2 Universal Database
# Attribute Definitions
#
# bin -> binary
# ces -> case exact string
# cis -> case insensitive string
# dn -> distinguished name
#
########################################################################

attribute binProperty                    1.3.18.0.2.4.305     bin
attribute binPropertyType                1.3.18.0.2.4.306     cis
attribute cesProperty                    1.3.18.0.2.4.307     ces
attribute cesPropertyType                1.3.18.0.2.4.308     cis
attribute cisProperty                    1.3.18.0.2.4.309     cis
attribute cisPropertyType                1.3.18.0.2.4.310     cis
attribute propertyType                   1.3.18.0.2.4.320     cis
attribute systemName                     1.3.18.0.2.4.329     cis
attribute db2nodeName                    1.3.18.0.2.4.419     cis
attribute db2nodeAlias                   1.3.18.0.2.4.420     cis
attribute db2instanceName                1.3.18.0.2.4.428     cis
attribute db2Type                        1.3.18.0.2.4.418     cis
attribute db2databaseName                1.3.18.0.2.4.421     cis
attribute db2databaseAlias               1.3.18.0.2.4.422     cis
attribute db2nodePtr                     1.3.18.0.2.4.423     dn
attribute db2gwPtr                       1.3.18.0.2.4.424     dn
attribute db2additionalParameters        1.3.18.0.2.4.426     cis
attribute db2ARLibrary                   1.3.18.0.2.4.427     cis
attribute db2authenticationLocation      1.3.18.0.2.4.425     cis
attribute db2databaseRelease             1.3.18.0.2.4.429     cis
attribute DCEPrincipalName               1.3.18.0.2.4.443     cis
```

以下のようにして、 DB2 オブジェクト・クラスを slapd.user_oc.conf ファイルに追加する必要があります。

```
########################################################################
#
# IBM DB2 Universal Database
# Object Class Definitions
```

```
#
###############################################################################

objectclass eProperty
        oid 1.3.18.0.2.6.90
        requires
                objectClass
        allows
                cn,
                propertyType,
                binProperty,
                binPropertyType,
                cesProperty,
                cesPropertyType,
                cisProperty,
                cisPropertyType

objectclass eApplicationSystem
        oid 1.3.18.0.2.6.84
        requires
                objectClass,
                systemName


objectclass DB2Node
        oid 1.3.18.0.2.6.116
        requires
                objectClass,
                db2nodeName
        allows
                db2nodeAlias,
                host,
                db2instanceName,
                db2Type,
                description,
                protocolInformation

objectclass DB2Database
        oid 1.3.18.0.2.6.117
        requires
                objectClass,
                db2databaseName,
                db2nodePtr
        allows
                db2databaseAlias,
                description,
                db2gwPtr,
                db2additionalParameters,
                db2authenticationLocation,
                DCEPrincipalName,
                db2databaseRelease,
                db2ARLibrary
```

DB2 スキーマ定義を追加した後、すべての変更を有効にするために、 Directory Server を再始動する必要があります。

関連資料:

- 368 ページの『DB2 で使用する LDAP オブジェクト・クラスおよび属性』

IBM SecureWay Directory Server 用にディレクトリー・スキーマを拡張する

IBM® SecureWay® LDAP Directory Server バージョン 3.1 以降を使用している場合、バージョン 8.2 より前の DB2® Universal Database (DB2 UDB) によって必要とされるオブジェクト・クラスおよび属性はすべて、基本スキーマに含まれます。バージョン 8.2 に含まれる新しい DB2 UDB 属性を使って基本スキーマを拡張するには、以下を実行します。

```
ldapmodify -c -h <machine_name>:389 -D <dn> -w <password> -f altgwnode.ldif
```

以下は、`altgwnode.ldif` ファイルの内容です。

```
dn:                cn=schema
changetype:        modify
add:               attributetypes
attributetypes:    (
 1.3.18.0.2.4.3092
 NAME 'db2altgwPtr'
 DESC 'DN pointer to DB2 alternate gateway (node) object'
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.12)
-
add:               ibmattributetypes
ibmattributetypes: (
 1.3.18.0.2.4.3092
 DBNAME ('db2altgwPtr' 'db2altgwPtr')
 ACCESS-CLASS NORMAL
 LENGTH 1000)


dn:                cn=schema
changetype:        modify
add:               attributetypes
attributetypes:    (
 1.3.18.0.2.4.3093
 NAME 'db2altnodePtr'
 DESC 'DN pointer to DB2 alternate node object'
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.12)
-
add:               ibmattributetypes
ibmattributetypes: (
 1.3.18.0.2.4.3093
 DBNAME ('db2altnodePtr' 'db2altnodePtr')
 ACCESS-CLASS NORMAL
 LENGTH 1000)


dn:            cn=schema
changetype:    modify
replace:       objectclasses
objectclasses: (
 1.3.18.0.2.6.117
 NAME 'DB2Database'
 DESC 'DB2 database'
 SUP cimSetting
 MUST ( db2databaseName $ db2nodePtr )
 MAY ( db2additionalParameters $ db2altgwPtr $ db2altnodePtr $ db2ARLibrary $ db2authenticationLocation $ db2databaseAlias $ db2databaseRelease $ db2gwPtr $ DCEPrincipalName ) )
```

altgwnode.ldif および altgwnode.readme ファイルは、
ftp://ftp.software.ibm.com/ps/products/db2/tools/ldap の URL にあります。

DB2 スキーマ定義を追加した後、すべての変更を有効にするために、 Directory Server を再始動する必要があります。

関連概念:

- 358 ページの『DB2 オブジェクト・クラスおよび属性を使用して LDAP ディレクトリー・スキーマを拡張する』
- 365 ページの『Sun One Directory Server 用にディレクトリー・スキーマを拡張する』

関連タスク:

- 358 ページの『Active Directory 用にディレクトリー・スキーマを拡張する』

Sun One Directory Server 用にディレクトリー・スキーマを拡張する

Sun One Directory Server は、 Netscape または iPlanet ディレクトリー・サーバーとしても知られています。

ご使用の環境で Sun One Directory Server を利用するには、 60ibmdb2.ldif ファイルを以下のディレクトリー・サーバーに追加してください。

Windows® では、 iPlanet が C:¥iPlanet¥Servers にインストール済みの場合、上記のファイルを .¥sldap-<machine_name>¥config¥schema に追加します。

UNIX® では、 iPlanet が /usr/iplanet/servers にインストール済みの場合、上記のファイルを ./slapd-<machine_name>/config/schema に追加します。

以下は、このファイルの内容です。

```
########################################################################
# IBM&regtm; DB2&regtm; Universal Database
########################################################################
dn: cn=schema
########################################################################
# Attribute Definitions (Before V8.2)
########################################################################
attributetypes: ( 1.3.18.0.2.4.305  NAME 'binProperty'               SYNTAX 1.3.6.1.4.1.1466.115.121.1.5               X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.306  NAME 'binPropertyType'           SYNTAX 1.3.6.1.4.1.1466.115.121.1.15              X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.307  NAME 'cesProperty'               SYNTAX 1.3.6.1.4.1.1466.115.121.1.15              X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.308  NAME 'cesPropertyType'           SYNTAX 1.3.6.1.4.1.1466.115.121.1.15              X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.309  NAME 'cisProperty'               SYNTAX 1.3.6.1.4.1.1466.115.121.1.15              X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.310  NAME 'cisPropertyType'           SYNTAX 1.3.6.1.4.1.1466.115.121.1.15              X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.320  NAME 'propertyType'              SYNTAX 1.3.6.1.4.1.1466.115.121.1.15              X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.329  NAME 'systemName'                SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.419  NAME 'db2nodeName'               SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.420  NAME 'db2nodeAlias'              SYNTAX 1.3.6.1.4.1.1466.115.121.1.15              X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.428  NAME 'db2instanceName'           SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.418  NAME 'db2Type'                   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.421  NAME 'db2databaseName'           SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.422  NAME 'db2databaseAlias'          SYNTAX 1.3.6.1.4.1.1466.115.121.1.15              X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.426  NAME 'db2additionalParameters'   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.427  NAME 'db2ARLibrary'              SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.425  NAME 'db2authenticationLocation' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.429  NAME 'db2databaseRelease'        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.443  NAME 'DCEPrincipalName'          SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.423  NAME 'db2nodePtr'                SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.424  NAME 'db2gwPtr'                  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
########################################################################
# Attribute Definitions (V8.2)
########################################################################
attributetypes: ( 1.3.18.0.2.4.3092 NAME 'db2altgwPtr'               SYNTAX 1.3.6.1.4.1.1466.115.121.1.12              X-ORIGIN 'IBM DB2' )
attributetypes: ( 1.3.18.0.2.4.3093 NAME 'db2altnodePtr'             SYNTAX 1.3.6.1.4.1.1466.115.121.1.12              X-ORIGIN 'IBM DB2' )
########################################################################
# Object Class Definitions
# DB2Database for V8.2 has the above two new optional attributes.
########################################################################
objectClasses: ( 1.3.18.0.2.6.90  NAME 'eProperty'          SUP top STRUCTURAL
 MAY ( cn $ propertyType $ binProperty $ binPropertyType $ cesProperty $ cesPropertyType $ cisProperty $ cisPropertyType )
 X-ORIGIN 'IBM DB2' )
objectClasses: ( 1.3.18.0.2.6.84  NAME 'eApplicationSystem' SUP top STRUCTURAL MUST systemName
 X-ORIGIN 'IBM DB2' )
objectClasses: ( 1.3.18.0.2.6.116 NAME 'DB2Node'            SUP top STRUCTURAL MUST db2nodeName
 MAY ( db2instanceName $ db2nodeAlias $ db2Type $ description $ host $ protocolInformation )
 X-ORIGIN 'IBM DB2' )
```

```
objectClasses: ( 1.3.18.0.2.6.117 NAME 'DB2Database'      SUP top STRUCTURAL MUST (db2databaseName $ db2nodePtr )
     MAY ( db2additionalParameters $ db2altgwPtr $ db2altnodePtr $ db2ARLibrary $ db2authenticationLocation $ db2databaseAlias $ db2databaseRelease $ db2gwPtr $ DCEPrincipalName $ description )
          X-ORIGIN 'IBM DB2' )
```

60ibmdb2.ldif および 60ibmdb2.readme ファイルは、ftp://ftp.software.ibm.com/ps/products/db2/tools/ldap の URL にあります。

DB2 スキーマ定義を追加した後、すべての変更を有効にするために、 Directory Server を再始動する必要があります。

関連概念:

- 358 ページの『DB2 オブジェクト・クラスおよび属性を使用して LDAP ディレクトリー・スキーマを拡張する』
- 363 ページの『IBM SecureWay Directory Server 用にディレクトリー・スキーマを拡張する』

関連タスク:

- 358 ページの『Active Directory 用にディレクトリー・スキーマを拡張する』

DB2 で使用する LDAP オブジェクト・クラスおよび属性

以下の表は、DB2 で使用するオブジェクト・クラスを示しています。

表 23. cimManagedElement

クラス	cimManagedElement
Active Directory LDAP 表示名	適用されない
Active Directory 共通名 (cn)	適用されない
説明	IBM Schema で使用される多くのシステム管理オブジェクト・クラスの基本クラスを提供します。
SubClassOf	最上位
必須属性	
オプション属性	説明
タイプ	抽象型
OID (オブジェクト ID)	1.3.18.0.2.6.132
GUID (グローバル・ユニーク ID)	b3afd63f-5c5b-11d3-b818-002035559151

表 24. cimSetting

クラス	cimSetting
Active Directory LDAP 表示名	適用されない
Active Directory 共通名 (cn)	適用されない
説明	IBM Schema での構成および設定に使用する基本クラスを提供します。
SubClassOf	cimManagedElement
必須属性	
オプション属性	settingID
タイプ	抽象型
OID (オブジェクト ID)	1.3.18.0.2.6.131
GUID (グローバル・ユニーク ID)	b3afd64d-5c5b-11d3-b818-002035559151

表 25. *eProperty*

クラス	**eProperty**
Active Directory LDAP 表示名	ibm-eProperty
Active Directory 共通名 (cn)	ibm-eProperty
説明	ユーザーが選択できるプロパティーに、アプリケーション特有の設定を指定するときに使います。
SubClassOf	cimSetting
必須属性	
オプション属性	propertyType cisPropertyType cisProperty cesPropertyType cesProperty binPropertyType binProperty
タイプ	構造
OID (オブジェクト ID)	1.3.18.0.2.6.90
GUID (グローバル・ユニーク ID)	b3afd69c-5c5b-11d3-b818-002035559151

表 26. *DB2Node*

クラス	**DB2Node**
Active Directory LDAP 表示名	ibm-db2Node
Active Directory 共通名 (cn)	ibm-db2Node
説明	DB2 サーバーを表します。
SubClassOf	eSap / ServiceConnectionPoint
必須属性	db2nodeName
オプション属性	db2nodeAlias db2instanceName db2Type host / dNSHostName (注 2 参照) protocolInformation/ServiceBindingInformation
タイプ	構造
OID (オブジェクト ID)	1.3.18.0.2.6.116
GUID (グローバル・ユニーク ID)	b3afd65a-5c5b-11d3-b818-002035559151

表 26. DB2Node (続き)

クラス	DB2Node
特別な注意事項	1. *DB2Node* クラスは、 IBM SecureWay ディレクトリーの下にある *eSap* オブジェクト・クラスと、 Microsoft Active Directory の下にある *ServiceConnectionPoint* オブジェクト・クラスから派生します。 2. *host* は IBM SecureWay 環境で使用されます。 *dNSHostName* 属性は Microsoft Active Directory で使用されます。 3. *protocolInformation* は、 IBM SecureWay 環境でのみ使用されます。 Microsoft Active Directory の場合は、 ServiceConnectionPoint クラスを継承する *ServiceBindingInformation* がプロトコル情報を収容するために使用されます。

DB2Node オブジェクトの *protocolInformation* 属性 (IBM SecureWay Directory で使用)、または *ServiceBindingInformation* 属性 (Microsoft Active Directory で使用) には、 DB2 データベース・サーバーにバインドするための通信プロトコル情報が格納されます。これは、サポートされているネットワーク・プロトコルを記述するトークンから成ります。各トークンはセミコロンで区切られます。トークンとトークンの間にスペースはありません。アスタリスク (*) を使用して、任意指定パラメーターを指定できます。

TCP/IP のトークンは以下のとおりです。

- ″TCPIP″
- サーバーのホスト名または IP アドレス
- サービス名 (svcename) またはポート番号 (例: 50000)
- (任意指定) セキュリティー (″NONE″ または ″SOCKS″)

APPN のトークンは以下のとおりです。

- ″APPN″
- ネットワーク ID
- パートナー LU
- トランザクション・プログラム (TP) 名 (Support Application TP のみ、 Service TP - TP in HEX はサポートしない)
- モード
- セキュリティー (″NONE″、″SAME″、または ″PROGRAM″ のいずれか)
- (任意指定) LAN アダプターのアドレス
- (任意指定) パスワード変更 LU

注: DB2 UDB for Windows クライアントでは、 APPN 情報がローカル SNA スタックで構成されておらず、かつ LAN アダプター・アドレスと任意指定の変更パスワード LU が LDAP にある場合、その DB2 UDB クライアントはこの情報を使用して SNA スタックを構成しようとします (スタックの構成方法が分かっている場合)。

NetBIOS のトークンは以下のとおりです。

- ″NETBIOS″
- サーバー NetBIOS のワークステーション名

名前付きパイプのトークンは以下のとおりです。

- ″NPIPE″
- サーバーのコンピューター名
- サーバーのインスタンス名

表 27. DB2Database

クラス	DB2Database
Active Directory LDAP 表示名	ibm-db2Database
Active Directory 共通名 (cn)	ibm-db2Database
説明	DB2 データベースを表します。
SubClassOf	最上位
必須属性	db2databaseName db2nodePtr
オプション属性	db2databaseAlias db2additionalParameter db2ARLibrary db2authenticationLocation db2gwPtr db2databaseRelease DCEPrincipalName db2altgwPtr db2altnodePtr
タイプ	構造
OID (オブジェクト ID)	1.3.18.0.2.6.117
GUID (グローバル・ユニーク ID)	b3afd659-5c5b-11d3-b818-002035559151

表 28. db2additionalParameters

属性	db2additionalParameters
Active Directory LDAP 表示名	ibm-db2AdditionalParameters
Active Directory 共通名 (cn)	ibm-db2AdditionalParameters
説明	ホスト・データベース・サーバーへの接続時に使用する、任意の追加パラメーターを含んでいます。
構文	大文字小文字を無視したストリング
最大の長さ	1024
複合値	単一値
OID (オブジェクト ID)	1.3.18.0.2.4.426

表 28. db2additionalParameters (続き)

属性	db2additionalParameters
GUID (グローバル・ユニーク ID)	b3afd315-5c5b-11d3-b818-002035559151

表 29. db2authenticationLocation

属性	db2authenticationLocation
Active Directory LDAP 表示名	ibm-db2AuthenticationLocation
Active Directory 共通名 (cn)	ibm-db2AuthenticationLocation
説明	認証が行われる場所を指定します。
構文	大文字小文字を無視したストリング
最大の長さ	64
複合値	単一値
OID (オブジェクト ID)	1.3.18.0.2.4.425
GUID (グローバル・ユニーク ID)	b3afd317-5c5b-11d3-b818-002035559151
注	有効な値は、CLIENT、SERVER、DCS、DCE、KERBEROS、SVRENCRYPT、または DCSENCRYPT です。

表 30. db2ARLibrary

属性	db2ARLibrary
Active Directory LDAP 表示名	ibm-db2ARLibrary
Active Directory 共通名 (cn)	ibm-db2ARLibrary
説明	アプリケーション・リクエスター・ライブラリーの名前
構文	大文字小文字を無視したストリング
最大の長さ	256
複合値	単一値
OID (オブジェクト ID)	1.3.18.0.2.4.427
GUID (グローバル・ユニーク ID)	b3afd316-5c5b-11d3-b818-002035559151

表 31. db2databaseAlias

属性	db2databaseAlias
Active Directory LDAP 表示名	ibm-db2DatabaseAlias
Active Directory 共通名 (cn)	ibm-db2DatabaseAlias
説明	データベース別名
構文	大文字小文字を無視したストリング
最大の長さ	1024
複合値	複合値
OID (オブジェクト ID)	1.3.18.0.2.4.422
GUID (グローバル・ユニーク ID)	b3afd318-5c5b-11d3-b818-002035559151

表 32. *db2databaseName*

属性	db2databaseName
Active Directory LDAP 表示名	ibm-db2DatabaseName
Active Directory 共通名 (cn)	ibm-db2DatabaseName
説明	データベース名
構文	大文字小文字を無視したストリング
最大の長さ	1024
複合値	単一値
OID (オブジェクト ID)	1.3.18.0.2.4.421
GUID (グローバル・ユニーク ID)	b3afd319-5c5b-11d3-b818-002035559151

表 33. *db2databaseRelease*

属性	db2databaseRelease
Active Directory LDAP 表示名	ibm-db2DatabaseRelease
Active Directory 共通名 (cn)	ibm-db2DatabaseRelease
説明	データベース・リリース番号
構文	大文字小文字を無視したストリング
最大の長さ	64
複合値	単一値
OID (オブジェクト ID)	1.3.18.0.2.4.429
GUID (グローバル・ユニーク ID)	b3afd31a-5c5b-11d3-b818-002035559151

表 34. *db2nodeAlias*

属性	db2nodeAlias
Active Directory LDAP 表示名	ibm-db2NodeAlias
Active Directory 共通名 (cn)	ibm-db2NodeAlias
説明	ノード別名
構文	大文字小文字を無視したストリング
最大の長さ	1024
複合値	複合値
OID (オブジェクト ID)	1.3.18.0.2.4.420
GUID (グローバル・ユニーク ID)	b3afd31d-5c5b-11d3-b818-002035559151

表 35. *db2nodeName*

属性	db2nodeName
Active Directory LDAP 表示名	ibm-db2NodeName
Active Directory 共通名 (cn)	ibm-db2NodeName
説明	ノード名
構文	大文字小文字を無視したストリング
最大の長さ	64
複合値	単一値
OID (オブジェクト ID)	1.3.18.0.2.4.419

表 35. db2nodeName (続き)

属性	db2nodeName
GUID (グローバル・ユニーク ID)	b3afd31e-5c5b-11d3-b818-002035559151

表 36. db2nodePtr

属性	db2nodePtr
Active Directory LDAP 表示名	ibm-db2NodePtr
Active Directory 共通名 (cn)	ibm-db2NodePtr
説明	データベースを所有しているデータベース・サーバーを表す、ノード (DB2Node) オブジェクトへのポインター。
構文	識別名
最大の長さ	1000
複合値	単一値
OID (オブジェクト ID)	1.3.18.0.2.4.423
GUID (グローバル・ユニーク ID)	b3afd31f-5c5b-11d3-b818-002035559151
特別な注意事項	この関連により、クライアントは、データベースへ接続するためのプロトコル通信情報を取り出すことができます。

表 37. db2altnodePtr

属性	db2altnodePtr
Active Directory LDAP 表示名	ibm-db2AltNodePtr
Active Directory 共通名 (cn)	ibm-db2AltNodePtr
説明	代替データベース・サーバーを表すノード (DB2Node) オブジェクトへのポインター。
構文	識別名
最大の長さ	1000
複合値	複合値
OID (オブジェクト ID)	1.3.18.0.2.4.3093
GUID (グローバル・ユニーク ID)	5694e266-2059-4e32-971e-0778909e0e72

表 38. db2gwPtr

属性	db2gwPtr
Active Directory LDAP 表示名	ibm-db2GwPtr
Active Directory 共通名 (cn)	ibm-db2GwPtr
説明	データベースへのアクセス元となるゲートウェイ・サーバーを表す、ノード・オブジェクトへのポインター。
構文	識別名
最大の長さ	1000
複合値	単一値
OID (オブジェクト ID)	1.3.18.0.2.4.424
GUID (グローバル・ユニーク ID)	b3afd31b-5c5b-11d3-b818-002035559151

表 39. *db2altgwPtr*

属性	db2altgwPtr
Active Directory LDAP 表示名	ibm-db2AltGwPtr
Active Directory 共通名 (cn)	ibm-db2AltGwPtr
説明	代替ゲートウェイ・サーバーを表すノード・オブジェクトへのポインター。
構文	識別名
最大の長さ	1000
複合値	複合値
OID (オブジェクト ID)	1.3.18.0.2.4.3092
GUID (グローバル・ユニーク ID)	70ab425d-65cc-4d7f-91d8-084888b3a6db

表 40. *db2instanceName*

属性	db2instanceName
Active Directory LDAP 表示名	ibm-db2InstanceName
Active Directory 共通名 (cn)	ibm-db2InstanceName
説明	データベース・サーバー・インスタンスの名前
構文	大文字小文字を無視したストリング
最大の長さ	256
複合値	単一値
OID (オブジェクト ID)	1.3.18.0.2.4.428
GUID (グローバル・ユニーク ID)	b3afd31c-5c5b-11d3-b818-002035559151

表 41. *db2Type*

属性	db2Type
Active Directory LDAP 表示名	ibm-db2Type
Active Directory 共通名 (cn)	ibm-db2Type
説明	データベース・サーバーのタイプ
構文	大文字小文字を無視したストリング
最大の長さ	64
複合値	単一値
OID (オブジェクト ID)	1.3.18.0.2.4.418
GUID (グローバル・ユニーク ID)	b3afd320-5c5b-11d3-b818-002035559151
注	有効なデータベース・サーバーのタイプは、SERVER、MPP、および DCS です。

表 42. *DCEPrincipalName*

属性	DCEPrincipalName
Active Directory LDAP 表示名	ibm-DCEPrincipalName
Active Directory 共通名 (cn)	ibm-DCEPrincipalName
説明	DCE プリンシパル名
構文	大文字小文字を無視したストリング

表 42. DCEPrincipalName (続き)

属性	DCEPrincipalName
最大の長さ	2048
複合値	単一値
OID (オブジェクト ID)	1.3.18.0.2.4.443
GUID (グローバル・ユニーク ID)	b3afd32d-5c5b-11d3-b818-002035559151

表 43. cesProperty

属性	cesProperty
Active Directory LDAP 表示名	ibm-cesProperty
Active Directory 共通名 (cn)	ibm-cesProperty
説明	この属性の値を使用すれば、アプリケーション固有の設定に関する構成パラメーターを提供できます。たとえば、値に XML 形式のデータを含めることができます。この属性の値は、 cesPropertyType 属性値ではすべて同じ種類でなければなりません。
構文	大文字小文字を区別するストリング
最大の長さ	32700
複合値	複合値
OID (オブジェクト ID)	1.3.18.0.2.4.307
GUID (グローバル・ユニーク ID)	b3afd2d5-5c5b-11d3-b818-002035559151

表 44. cesPropertyType

属性	cesPropertyType
Active Directory LDAP 表示名	ibm-cesPropertyType
Active Directory 共通名 (cn)	ibm-cesPropertyType
説明	この属性の値を使用すれば、 cesProperty 属性のすべての値の構文、意味、その他の特性を記述できます。たとえば、″XML″ という値を使用すると、 cesProperty 属性のすべての値が XML 構文としてエンコードされることを示します。
構文	大文字小文字を無視したストリング
最大の長さ	128
複合値	複合値
OID (オブジェクト ID)	1.3.18.0.2.4.308
GUID (グローバル・ユニーク ID)	b3afd2d6-5c5b-11d3-b818-002035559151

表 45. cisProperty

属性	cisProperty
Active Directory LDAP 表示名	ibm-cisProperty
Active Directory 共通名 (cn)	ibm-cisProperty

表 45. *cisProperty (続き)*

属性	cisProperty
説明	この属性の値を使用すれば、アプリケーション固有の設定に関する構成パラメーターを提供できます。たとえば、値に INI ファイルを含めることができます。この属性の値は、 cisPropertyType 属性値ではすべて同じ種類でなければなりません。
構文	大文字小文字を無視したストリング
最大の長さ	32700
複合値	複合値
OID (オブジェクト ID)	1.3.18.0.2.4.309
GUID (グローバル・ユニーク ID)	b3afd2e0-5c5b-11d3-b818-002035559151

表 46. *cisPropertyType*

属性	cisPropertyType
Active Directory LDAP 表示名	ibm-cisPropertyType
Active Directory 共通名 (cn)	ibm-cisPropertyType
説明	この属性の値を使用すれば、 cisProperty 属性のすべての値の構文、意味、その他の特性を記述できます。たとえば、″INI File″ という値を使用すると、 cisProperty 属性のすべての値が INI ファイルであることを示します。
構文	大文字小文字を無視したストリング
最大の長さ	128
複合値	複合値
OID (オブジェクト ID)	1.3.18.0.2.4.310
GUID (グローバル・ユニーク ID)	b3afd2e1-5c5b-11d3-b818-002035559151

表 47. *binProperty*

属性	binProperty
Active Directory LDAP 表示名	ibm-binProperty
Active Directory 共通名 (cn)	ibm-binProperty
説明	この属性の値を使用すれば、アプリケーション固有の設定に関する構成パラメーターを提供できます。たとえば、値にバイナリーでエンコードされた Lotus 1-2-3 のプロパティーを含めることができます。この属性の値は、 binPropertyType 属性値ではすべて同じ種類でなければなりません。
構文	バイナリー
最大の長さ	250000
複合値	複合値
OID (オブジェクト ID)	1.3.18.0.2.4.305
GUID (グローバル・ユニーク ID)	b3afd2ba-5c5b-11d3-b818-002035559151

表 48. *binPropertyType*

属性	**binPropertyType**
Active Directory LDAP 表示名	ibm-binPropertyType
Active Directory 共通名 (cn)	ibm-binPropertyType
説明	この属性の値を使用すれば、 binProperty 属性のすべての値の構文、意味、その他の特性を記述できます。たとえば、″Lotus 123″ という値を使用すると、 binProperty 属性のすべての値がバイナリーでエンコードされた Lotus 1-2-3 のプロパティーであることを示します。
構文	大文字小文字を無視したストリング
最大の長さ	128
複合値	複合値
OID (オブジェクト ID)	1.3.18.0.2.4.306
GUID (グローバル・ユニーク ID)	b3afd2bb-5c5b-11d3-b818-002035559151

表 49. *PropertyType*

属性	**PropertyType**
Active Directory LDAP 表示名	ibm-propertyType
Active Directory 共通名 (cn)	ibm-propertyType
説明	この属性の値は、 eProperty オブジェクトの意味特性を記述します。
構文	大文字小文字を無視したストリング
最大の長さ	128
複合値	複合値
OID (オブジェクト ID)	1.3.18.0.2.4.320
GUID (グローバル・ユニーク ID)	b3afd4ed-5c5b-11d3-b818-002035559151

表 50. *settingID*

属性	**settingID**
Active Directory LDAP 表示名	適用されない
Active Directory 共通名 (cn)	適用されない
説明	命名属性。 eProperty などの cimSetting から派生したオブジェクト項目を示すために使用できます。
構文	大文字小文字を無視したストリング
最大の長さ	256
複合値	単一値
OID (オブジェクト ID)	1.3.18.0.2.4.325
GUID (グローバル・ユニーク ID)	b3afd596-5c5b-11d3-b818-002035559151

関連概念:

- 337 ページの『Lightweight Directory Access Protocol (LDAP) の紹介』

付録 D. 複数のデータベース・パーティションに対するコマンドの発行

パーティション・データベース環境でのコマンドの実行

パーティション・データベース・システムでは、インスタンスにあるマシンで、あるいはデータベース・パーティション・サーバー (ノード) で実行するコマンドを発行したい場合があります。このような場合には、**rah** コマンドまたは **db2_all** コマンドを使用することができます。 **rah** コマンドを使用すれば、インスタンス内のすべてのマシンで実行したいコマンドを発行できます。インスタンス内のデータベース・パーティション・サーバーでコマンドを実行したい場合は、 **db2_all** コマンドを実行します。この節では、これらのコマンドについての概要を説明します。以下の情報は、パーティション・データベース・システムだけに適用されます。

注:

1. UNIX® ベースのプラットフォームでは、ログイン・シェルを Korn シェルまたは他のシェルにすることができます。ただし、特殊文字を含むコマンドをシェルが処理する方法は、シェルによってさまざまに異なります。
2. Windows NT で **rah** コマンドまたは **db2_all** コマンドを実行するには、管理者グループのメンバーになっているユーザー・アカウントでログオンしなければなりません。

コマンドの有効範囲を調べるには、「コマンド・リファレンス」を参照してください。本書では、コマンドが 1 つのデータベース・パーティション・サーバーで実行されるか、それともすべてのサーバーで実行されるかを示します。 1 つのデータベース・パーティション・サーバーで実行されるコマンドをすべてのサーバーで実行させるには、 **db2_all** を使用してください。 **db2trc** コマンドは例外で、 1 つのマシン上のすべての論理ノード (データベース・パーティション・サーバー) で実行します。すべてのマシン上のすべての論理ノードで **db2trc** を実行したい場合は、**rah** を使用してください。

関連概念:

- 379 ページの『rah および db2_all コマンドの概要』
- 381 ページの『rah および db2_all コマンドの指定』

関連資料:

- 380 ページの『rah および db2_all コマンドの説明』

rah および db2_all コマンドの概要

コマンド群の実行は、1 つのデータベース・パーティション・サーバーから別のデータベース・パーティション・サーバーへと順次行うことも、並列に行うこともできます。 UNIX® ベースのプラットフォームでコマンドを並列に実行する場合は、出力をまとめて表示する (デフォルトの動作) か、コマンドが発行されるマシンに出

力を表示することができます。 Windows NT では、コマンドを並列に実行すると、出力はそのコマンドが発行されるマシンに表示されます。

rah コマンドを使用するには、次のように入力してください。

```
rah command
```

db2_all コマンドを使用するには、次のように入力してください。

```
db2_all command
```

rah 構文に関するヘルプを表示するには、次のように入力してください。

```
rah "?"
```

対話式プロンプトで入力できるほとんどのコマンドを使用できます。たとえば、複数のコマンドを順番に実行することも可能です。 UNIX ベースのプラットフォームでは、セミコロン (;) を使って複数のコマンドを分離します。 Windows NT では、アンパーサンド (&) を使って複数のコマンドを分離します。最後のコマンドの後には、区切り記号を使用しないでください。

以下の例は、**db2_all** コマンドを使用して、ノード構成ファイルで指定したすべてのデータベース・パーティションでデータベース構成を変更する方法を示したものです。
; 文字が二重引用符の内側にあるので、要求は同時に実行されます。

```
db2_all ";UPDATE DB CFG FOR sample USING LOGFILSIZ 100"
```

関連概念:

- 379 ページの『パーティション・データベース環境でのコマンドの実行』
- 381 ページの『rah および db2_all コマンドの指定』

関連資料:

- 380 ページの『rah および db2_all コマンドの説明』

rah および db2_all コマンドの説明

以下のコマンドを使用できます。

コマンド	説明
rah	すべてのマシンでコマンドを実行します。
db2_all	指定したすべてのデータベース・パーティション・サーバーでコマンドを実行します。
db2_kill	複数のデータベース・パーティション・サーバーで実行されているすべてのプロセスを突然停止し、すべてのデータベース・パーティション・サーバーのすべてのリソースを終結処理します。このコマンドは、データベースを不整合にします。このコマンドは IBM サービスからの指示による以外は発行しないでください。
db2_call_stack	UNIX ベースのプラットフォームでは、すべてのデータベース・パーティション・サーバーで実行されているすべてのプロセスが、呼び出しトレースバックを syslog に書き出すようにします。

Windows NT では、すべてのデータベース・パーティション・サーバーで実行されているすべてのプロセスが、呼び出しトレースバックをインスタンス・ディレクトリー内の P*xxxx.nnn* ファイルに書き出すようにします (P*xxxx* はプロセス ID、*nnn* はノード番号)。

UNIX ベースのプラットフォームでは、これらのコマンドは、次のような特定の暗黙的な設定で **rah** を実行します。

- すべてのマシンで並列に実行する。
- コマンド出力を `/tmp/$USER/db2_kill`、 `/tmp/$USER/db2_call_stack` にそれぞれバッファーする。

Windows NT では、これらのコマンドは **rah** を実行して、すべてのマシンで並列に実行します。

関連概念:

- 379 ページの『rah および db2_all コマンドの概要』
- 381 ページの『rah および db2_all コマンドの指定』
- 382 ページの『UNIX ベース・プラットフォームでのコマンドの並列実行』

rah および db2_all コマンドの指定

次のようにコマンドを指定することができます。

- パラメーターとして、コマンド行から。
- 何もパラメーターを指定しないときは、プロンプトに対する応答として。

コマンドに次のような特殊文字が入っている場合は、プロンプト方式を使用する必要があります。

```
| & ; < > ( ) { } [ ] unsubstituted $
```

コマンド行でパラメーターとしてコマンドを指定するときに、上にリストしたような特殊文字のいずれかが含まれている場合は、二重引用符で囲む必要があります。

注: UNIX® ベースのプラットフォームでは、プロンプトで入力した場合と同様に、コマンドがコマンド履歴に追加されます。

コマンドの中の特殊文字はすべて、正常に入力することができます (¥ 以外は引用符で囲まずに)。コマンドに ¥ を入れる必要があるときは、円記号を 2 つ (¥¥) 入力しなければなりません。

注: UNIX ベースのプラットフォームでは、Korn シェルを使用していない場合は、コマンドの中の特殊文字はすべて正常に入力することができます ("、¥、置換不能文字 $、および単一引用符 (') 以外は、引用符に入れずに)。 コマンドにこれらの文字のいずれかを入れる必要があるときは、円記号を 3 つ (¥¥¥) 前に置かなければなりません。たとえば、コマンドに ¥ を入れる必要があるときは、円記号を 4 つ (¥¥¥¥) 入力しなければなりません。

コマンドに 2 重引用符 (") を入れる必要があるときは、円記号を 3 つ前に付けて、たとえば、¥¥¥" のように入力しなければなりません。

注:

1. UNIX ベースのプラットフォームでは、単一引用符付きストリングの内側に単一引用符を入れる何らかの方法をコマンド・シェルが提供しないかぎり、単一引用符 (') をコマンドに含めることはできません。
2. Windows NT では、単一引用符付きストリングの内側に単一引用符を入れる何らかの方法をコマンド・ウィンドウが提供しないかぎり、コマンドに単一引用符 (') を含めることはできません。

stdin からバックグラウンドで読み取りを行うロジックを含む Korn シェルのシェル・スクリプトを実行する場合、 stdin をソースに明示的にリダイレクトする必要があります。そうすれば、プロセスは端末上で停止されることなく読み取りを行うことができます (SIGTTIN メッセージ)。 stdin をリダイレクトするには、指定されている入力がなければ次の形式のスクリプトを実行します。

```
shell_script </dev/null &
```

同様に、db2_all をバックグラウンドで実行する際には、常に </dev/null を指定する必要があります。たとえば、次のようなものがあります。

```
db2_all ";run_this_command" </dev/null &
```

これを行うことにより、端末上で停止させなくても stdin をリダイレクトすることができます。

別の方法として、リモート・コマンドからの出力が必要ない場合には、次のように db2_all 接頭部でデーモン化オプションを使用することもできます。

```
db2_all ";daemonize_this_command" &
```

関連概念:

- 382 ページの『UNIX ベース・プラットフォームでのコマンドの並列実行』
- 384 ページの『その他の rah (Solaris および AIX のみ)』

関連タスク:

- 391 ページの『Windows NT での rah のデフォルト環境プロファイルの設定』

関連資料:

- 380 ページの『rah および db2_all コマンドの説明』
- 385 ページの『rah コマンドの接頭部シーケンス』
- 389 ページの『rah コマンドの制御』

UNIX ベース・プラットフォームでのコマンドの並列実行

注: この節の情報は UNIX® ベースのプラットフォームだけに適用されます。

デフォルトでは、コマンドはそれぞれのマシンで順次的に実行されますが、特定の接頭部シーケンスをコマンドの前につけることによって、バックグラウンド rshell を使って、コマンドを並列に実行するよう指定できます。 rshell がバックグラウンドで実行されている場合、それぞれのコマンドは、リモート・マシンにあるバッファー・ファイルに出力を入れます。このプロセスでは、出力は次のように 2 つに分けて取り出されます。

1. リモート・コマンドが完了した後。
2. 何らかのプロセスがまだ実行されている場合は、あとで実行される可能性のある rshell が終了した後。

デフォルトでは、バッファー・ファイルの名前は /tmp/$USER/rahout ですが、環境変数 $RAHBUFDIR/$RAHBUFNAME によって名前を指定することができます。

複数のコマンドを並行して実行するよう指定した場合、デフォルトでこのスクリプトは、すべてのホストにコマンドを送信する前に、 $RAHBUFDIR と $RAHBUFNAME で指定されるバッファー・ファイルが使用できるかどうかチェックします。存在していない場合、$RAHBUFDIR を作成します。この動作を省略するには、環境変数 RAHCHECKBUF=no を設定します。ディレクトリーが存在していて、使用可能であることがわかっている場合は、このようにすると時間を節約できます。

rah を使用して複数のマシンでコマンドを同時に実行する前に、以下を行ってください。

- それぞれのマシンごとに、使用しているユーザー ID 用のディレクトリー /tmp/$USER が存在することを確認する。このディレクトリーがまだ存在していない場合は、それを作成するために以下を実行してください。

  ```
  rah ")mkdir /tmp/$USER"
  ```

- 以下の行を .kshrc (Korn シェル構文の場合) または .profile に追加して、現行のセッションにそれを入力する。

  ```
  export RAHCHECKBUF=no
  ```

- リモート・コマンドを実行するそれぞれのマシン ID の .rhosts ファイル内に、**rah** を実行する ID に対応する項目があることを確認する。および **rah** を実行する ID の .rhosts ファイル内に、リモート・コマンドを実行するそれぞれのマシン ID に対応する項目があることを確認する。

関連概念:

- 384 ページの『その他の rah (Solaris および AIX のみ)』

関連タスク:

- 383 ページの『UNIX ベース・プラットフォームでの rah プロセスのモニター』

関連資料:

- 385 ページの『rah コマンドの接頭部シーケンス』
- 392 ページの『UNIX ベース・プラットフォームの場合の rah に関する問題の判別』

UNIX ベース・プラットフォームでの rah プロセスのモニター

手順:

注: この節の情報は UNIX ベースのプラットフォームだけに適用されます。
リモート・コマンドがまだ実行しているか、またはバッファー出力がまだ累積されている間は、 rah によって開始されたプロセスは、以下のような活動をモニターします。

- どのコマンドが実行されなかったかを示すメッセージを端末に書き出す。
- バッファー出力を検索する。

環境変数 RAHWAITTIME によって制御されるインターバルで、通知メッセージが書き出されます。この指定方法の詳細については、ヘルプ情報を参照してください。環境変数 RAHWAITTIME=0 を設定することににによって、すべての通知メッセージを完全に抑止できます。

1 次モニター・プロセスは、 **rahwaitfor** という名前のコマンド (ps コマンドで示される) です。 最初の通知メッセージで、このプロセスの pid (プロセス ID) が示されます。その他のすべてのモニター・プロセスは、 **rah** スクリプト (またはシンボリック・リンクの名前) を実行する **ksh** コマンドとして表示されます。必要であれば、次のコマンドによって、すべてのモニター・プロセスを停止することができます。

```
kill <pid>
```

ここで、<pid> は、1 次モニター・プロセスのプロセス ID です。シグナル番号を指定してはなりません。デフォルトである 15 のままにしてください。これは、リモート・コマンドにはまったく影響しませんが、バッファー出力を自動的に表示しないようにします。 **rah** の 1 回の実行中に、 2 つ以上の異なるモニター・プロセスのセットが、異なる時点で実行される可能性があることに注意してください。ただし、任意の時点で現行のセットを停止すると、その後はもう開始されません。

通常のログイン・シェルが /bin/ksh のような Korn シェルでない場合には、 **rah** を使用できますが、以下の特殊文字が含まれるコマンドを入力するときの規則が少し異なります。

```
"  unsubstituted $ '
```

詳細情報を表示するには、rah "?" と入力してください。また、UNIX ベースの環境では、リモート・コマンドを実行する ID のログイン・シェルが Korn シェルでない場合、 **rah** を実行する ID のログイン・シェルもまた、 Korn シェルであってはなりません。 (**rah** は、リモート ID のシェルが、ローカル ID に基づく Korn シェルであるかどうか判断します。) シェルは、単一引用符で囲まれたストリングに対して、置換または特別な処理を行ってはなりません。厳密に元のままにする必要があります。

関連概念:

- 382 ページの『UNIX ベース・プラットフォームでのコマンドの並列実行』
- 384 ページの『その他の rah (Solaris および AIX のみ)』

その他の rah (Solaris および AIX のみ)

パフォーマンスを向上させるために、 rah は大規模なシステムで tree_logic を使うように拡張されています。つまり、rah はリストに含まれるノード数を検査し、その数がしきい値を超過するなら、リストのサブセットを作成して、それ自体の再帰的呼び出しをそれぞれのノードに送信します。 それぞれのノードでは、再帰的に呼び出された rah は前述の同じ論理に従います。これは、リストが十分に小さくなり、「リスト上のすべてのノードにコマンドを送信する」という標準的な論理 (ここで

は ″ツリーのリーフ″ という論理) に従えるようになるまで続きます。このときのしきい値は、環境変数 RAHTREETHRESH で指定できます。これを指定しないと、デフォルトの 15 になります。

物理ノードに対して複数の論理ノードが存在するシステムの場合は、 db2_all は再帰的な呼び出しをそれぞれの物理ノードに送信してから、その同じ物理ノード上の他の論理ノードに rsh するので、物理ノード間の通信量も少なくなります。 (この点は、db2_all だけに当てはまるもので rah には当てはまりません。 rah は常にそれぞれの物理ノードだけに送信します。)

関連概念:

- 382 ページの『UNIX ベース・プラットフォームでのコマンドの並列実行』

関連タスク:

- 383 ページの『UNIX ベース・プラットフォームでの rah プロセスのモニター』

rah コマンドの接頭部シーケンス

接頭部シーケンスは、1 桁以上の特殊文字です。コマンドの文字の直前に、ブランクをあけずに 1 つまたは複数の接頭部シーケンスを入力してください。シーケンスを 2 つ以上指定したい場合は、任意の順序でタイプすることができますが、複数文字のシーケンスの中にある文字は、順番に入力する必要があります。接頭部シーケンスを入力するときは、次の例に示すように、接頭部シーケンスも含めてコマンド全体を二重引用符で囲んでください。

- UNIX ベースのプラットフォームでは、以下のようにします。

```
rah "};ps -F pid,ppid,etime,args -u $USER"
```

- Windows NT では、以下のようにします。

```
rah "||db2 get db cfg for sample"
```

接頭部シーケンスは次のとおりです。

シーケンス	目的
|	バックグラウンドでコマンドを順に実行します。
|&	バックグラウンドで順番にコマンドを実行し、さらにすべてのリモート・コマンドが完了したあとで、まだいくつかのプロセスが実行中であっても、コマンドを終了します。たとえば、子プロセス (UNIX ベースのプラットフォームの場合) またはバックグラウンド・プロセス (Windows の場合) がまだ実行中であれば、もっと後になる可能性があります。このケースでは、コマンドは、別個のバックグラウンド・プロセスを開始して、コマンド終了後に生成されたリモート出力を検索し、もとのマシンに書き戻します。 **注:** UNIX ベースのプラットフォームでは、& を指定すると、さらに **rsh** コマンドが必要になるので、パフォーマンスが低下します。
||	バックグラウンドでコマンドを並列に実行します。

||& バックグラウンドで並列にコマンドを実行し、さらにすべてのリモート・コマンドが完了した後で、上記の |& のケースで述べたようにしてコマンドを終了します。

注: UNIX ベースのプラットフォームでは、& を指定すると、さらに **rsh** コマンドが必要になるので、パフォーマンスが低下します。

; 上記の ||& と同じ。これは、省略形です。

注: ; を指定すると、 **rsh** コマンドがさらに必要になるので、 || に比べてパフォーマンスが低下します。

] コマンドを実行する前に、ユーザーのプロファイルのドット実行 (dot-execution(..)) を付加します。

注: UNIX ベースのプラットフォームに限り使用できます。

} コマンドを実行する前に、 $RAHENV (多分 `.kshrc`) で指名されたファイルのドット実行を付加します。

注: UNIX ベースのプラットフォームに限り使用できます。

]} コマンドを実行する前に、ユーザーのプロファイルのドット実行を付加し、その後で、 $RAHENV (多分 `.kshrc`) で指名されたファイルを実行します。

注: UNIX ベースのプラットフォームに限り使用できます。

) ユーザーのプロファイルおよび $RAHENV で指名されたファイルの実行を抑止します。

注: UNIX ベースのプラットフォームに限り使用できます。

' コマンドの起動をマシンにエコーします。

< このマシン以外のすべてのマシンに送信します。

<<-nnn< *nnn* 以外のすべてのデータベース・パーティション・サーバー (ノード番号 *nnn* を除いて `db2nodes.cfg` にあるすべてのデータベース・パーティション・サーバー。この表の最後の接頭部文字の後の最初の段落を参照してください) に送信します。

<<+nnn< データベース・パーティション・サーバー *nnn* (ノード番号が *nnn* の `db2nodes.cfg` にあるデータベース・パーティション・サーバー。この表の最後の接頭部文字の後の最初の段落を参照してください) だけに送信します。

(ブランク文字)

リモート・コマンドを、`stdin`、 `stdout` および `stderr` をすべてクローズしてバックグラウンドで実行します。このオプションは、バックグラウンドでコマンドを実行するときだけ、つまり ¥ または ; も含んでいる接頭部シーケンスの中でだけ有効です。このようにすると、コマンドは非常に早く完了することができます (リモート・コマンドが開始されるとすぐに)。この接頭部文字を **rah** コマンド

行で指定する場合は、コマンドを単一引用符で囲むか、またはコマンドを 2 重引用符で囲んでから接頭部文字の前に ¥ を置きます。たとえば、

```
rah '; mydaemon'
```

または

```
rah ";¥ mydaemon"
```

rah コマンドは、バックグラウンド・プロセスとして実行されるとき、出力が戻されるのを待ちません。

> <> のオカレンスをマシン名と置換します。

" () のオカレンスをマシン索引と置換し、 ## のオカレンスをノード番号と置換します。

注:

1. マシン索引とは、データベース・システムのマシンに関連した番号のことです。複数の論理ノードを実行している場合は、マシンのマシン索引は、ノード構成ファイル内のそのマシンのノード番号に対応します。複数の論理ノードを実行しているマシンの場合は、項目数もそのノードの数になるので、このようなマシンのマシン索引を取得するには、その重複項目をカウントしないでください。たとえば MACH1 と MACH2 の両方とも 2 つの論理ノードを実行している場合は、ノード構成ファイル内の MACH3 のノード番号は 5 になります。しかし、MACH3 のマシン索引は 3 になります。

   Windows NT の場合、ノード構成ファイルを編集しないでください。マシン索引を取得するには、 **db2nlist** コマンドを使用してください。
2. " が指定されている場合は、重複はマシンのリストから除去されません。

<<-nnn< と <<+nnn< 接頭部シーケンスを使用しているとき、 *nnn* は 1、2 または 3 桁の任意のパーティション番号にすることができますが、これは、 db2nodes.cfg ファイルの *nodenum* 値と一致している必要があります。

注: 接頭部シーケンスは、コマンドの一部と見なされます。接頭部シーケンスをコマンドの一部として指定するときは、接頭部シーケンスも含めてコマンド全体を 2 重引用符で囲んでください。

関連概念:

- 381 ページの『rah および db2_all コマンドの指定』
- 382 ページの『UNIX ベース・プラットフォームでのコマンドの並列実行』

関連資料:

- 380 ページの『rah および db2_all コマンドの説明』

区画に分割された環境でのマシンのリストの指定

手順:

デフォルトでは、マシンのリストは、ノード構成ファイル db2nodes.cfg から取得されます。以下のようにして、これをオーバーライドすることができます。

- 環境変数 RAHOSTFILE をエクスポート (UNIX ベースのプラットフォームの場合) または設定 (Windows NT の場合) することによって、マシンのリストが含まれるファイルのパス名を指定する。
- 環境変数 RAHOSTLIST をエクスポート (UNIX ベースのプラットフォームの場合) または設定 (Windows NT の場合) することによって、リストを明示的に、名前のストリングとしてスペースで区切って指定する。

  注: これらの環境変数が両方とも指定されている場合は、 RAHOSTLIST の方が優先します。

注: Windows NT の場合、ノード構成ファイル内で不整合が生じないようにするために、このファイルを編集しないでください。インスタンスにあるマシンのリストを取得するには、 **db2nlist** コマンドを使用してください。

関連タスク:

- 388 ページの『区画に分割された環境でのマシンのリストからの重複項目の除去』

区画に分割された環境でのマシンのリストからの重複項目の除去

手順:

1 つのマシンで複数の論理ノード (データベース・パーティション・サーバー) を使って DB2 Universal Database Enterprise Server Edition を実行している場合、db2nodes.cfg にはそのマシンに対する項目が複数含まれます。この状況では、各マシンにつき 1 回だけコマンドを実行するのか、それとも db2nodes.cfg ファイルにリストされている各論理ノードごとに 1 回ずつコマンドを実行するのかを、 **rah** コマンドが知っていなければなりません。マシンを指定するには **rah** コマンドを使用します。論理ノードを指定するには **db2_all** コマンドを使用します。

注: UNIX ベースのプラットフォームでは、マシンを指定した場合、通常、 **rah** はマシン・リストから重複を除去します。ただし、論理ノードを指定した場合、 **db2_all** は以下の割り当てをコマンドの前に付加します。

```
export DB2NODE=nnn  (Korn シェル構文の場合)
```

ここで、*nnn* は、目的のデータベース・パーティション・サーバーにコマンドを経路指定するために、 db2nodes.cfg ファイル内の対応する行から取得されるノード番号です。

論理ノードを指定するときには、 <<-nnn< および <<+nnn< 接頭部シーケンスを使って、それぞれ 1 つを除くすべての論理ノードが含まれるようにリストを限定したり、 1 つのデータベース・パーティション・サーバーだけを指定したりすることができます。これを行うのは、カタログ・ノードで最初にコマンドを実行して、完了

後に他のすべてのデータベース・パーティション・サーバーで (並列に) 同じコマンドを実行するような場合です。 **db2 restart database** コマンドを実行するときには、通常、これが必要です。これを行うには、カタログ・ノードのノード番号を知っておく必要があります。

rah コマンドを使用して **db2 restart database** を実行するとき、重複項目はマシンのリストから除去されます。しかし、" 接頭部を指定する場合、 " 接頭部は各マシンではなく各データベース・パーティション・サーバーに送信することを意味するため、重複は除去されません。

関連タスク:

- 388 ページの『区画に分割された環境でのマシンのリストの指定』

関連資料:

- 「コマンド・リファレンス」の『RESTART DATABASE コマンド』
- 385 ページの『rah コマンドの接頭部シーケンス』

rah コマンドの制御

rah コマンドを制御するために以下の環境変数を使用することができます。

表 51.

名前	意味	デフォルト
$RAHBUFDIR **注:** UNIX ベースのプラットフォームに限り使用できます。	バッファーのディレクトリー	/tmp/$USER
$RAHBUFNAME **注:** UNIX ベースのプラットフォームに限り使用できます。	バッファーのファイル名	rahout
$RAHOSTFILE (UNIX ベースのプラットフォームの場合)、RAHOSTFILE (Windows NT の場合)	ホストのリストが入っているファイル	db2nodes.cfg
$RAHOSTLIST (UNIX ベースのプラットフォームの場合)、RAHOSTLIST (Windows NT の場合)	ストリングとしてのホストのリスト	$RAHOSTFILE から抽出
$RAHCHECKBUF **注:** UNIX ベースのプラットフォームに限り使用できます。	″no″ に設定されていると、チェックはバイパスします。	設定されない

表 51. (続き)

名前	意味	デフォルト
$RAHSLEEPTIME (UNIX ベースのプラットフォームの場合)、RAHSLEEPTIME (Windows NT の場合)	並列に実行されるコマンドからの最初の出力をこのスクリプトが待つ時間 (秒数)。	**db2_kill** の場合は 86400 秒、すべての他の場合 200 秒。
$RAHWAITTIME (UNIX ベースのプラットフォームの場合)、RAHWAITTIME (Windows NT の場合)	Windows NT の場合、リモート・ジョブがまだ実行されていることを連続チェックするインターバルの秒数 UNIX ベースのプラットフォームの場合、リモート・ジョブがまだ実行されていることを連続チェックしてから、 `rah: waiting for <pid> ...` メッセージまでのインターバルの秒数 プラットフォームに関係なく、正の整数を指定します。メッセージを抑止するには先行ゼロを値の前に付けます。たとえば、RAHWAITTIME=045 を指定してエクスポートします。 **rah** は、ジョブの完了を検知するためにこれらのチェックには依存しないので、低い値を指定する必要はありません。	45 秒
$RAHENV **注:** UNIX ベースのプラットフォームに限り使用できます。	$RAHDOTFILES=E または K または PE または B の場合に実行されるファイル名を指定します。	$ENV
$RAHUSER (UNIX ベースのプラットフォームの場合)、RAHUSER (Windows NT の場合)	UNIX ベースのプラットフォームの場合、リモート・コマンドがその下で実行されるユーザー ID Windows NT の場合、DB2 リモート・コマンド・サービスに関連したログオン・アカウント	$USER

注: UNIX ベースのプラットフォームでは、リモート・シェルで設定される値 (存在する場合) ではなく、 **rah** が実行される $RAHENV の値が使用されます。

関連資料:

- 390 ページの『UNIX ベース・プラットフォームでの $RAHDOTFILES の使用』

UNIX ベース・プラットフォームでの $RAHDOTFILES の使用

注: この節の情報は UNIX ベースのプラットフォームだけに適用されます。

接頭部シーケンスが指定されていない場合に実行される .files は、次のとおりです。

P .profile

E $RAHENV で指名されたファイル (通常は .kshrc)

K E と同じ

PE $RAHENV で指名されたファイル (通常は .kshrc) が後に続く .profile

B PE と同じ

N なし (またはどちらでもない)

注: ログイン・シェルが Korn シェルでない場合は、実行を指定した任意のドット・ファイルは Korn シェル・プロセスで実行されるので、 Korn シェル構文に従う必要があります。したがって、たとえばログイン・シェルが C シェルである場合、 **rah** によって実行されるコマンド用に .cshrc 環境をセットアップするために、その .cshrc に相応する Korn シェル INSTHOME/.profile を作成して、 INSTHOME/.cshrc の中で指定する必要があります。

```
setenv RAHDOTFILES P
```

あるいは、.cshrc に相応する Korn シェル INSTHOME/.kshrc を作成して、INSTHOME/.cshrc の中で指定する必要があります。

```
setenv RAHDOTFILES E
setenv RAHENV INSTHOME/.kshrc
```

また、tty が存在しない場合 (**rsh** によって起動される場合など) には、 .cshrc を stdout に書き出さないようにすることも重要です。そうするには、stdout に書き出す行を、たとえば次のように囲むことができます。

```
if { tty -s } then echo "executed .cshrc";
endif
```

関連資料:

- 389 ページの『rah コマンドの制御』

Windows NT での rah のデフォルト環境プロファイルの設定

手順:

注: この節の情報は Windows NT だけに適用されます。
rah コマンドのデフォルト環境プロファイルを設定するには、 db2rah.env ファイルを使用します。これはインスタンス・ディレクトリー内に作成する必要があります。ファイルは以下の形式にする必要があります。

```
; This is a comment line
DB2INSTANCE=instancename
DB2DBDFT=database
; End of file
```

rah の環境を初期設定するのに必要な環境変数をすべて指定できます。

関連概念:

- 381 ページの『rah および db2_all コマンドの指定』

UNIX ベース・プラットフォームの場合の rah に関する問題の判別

注: この節の情報は UNIX ベースのプラットフォームだけに適用されます。
rah を実行しているときに検出される可能性がある問題の処理方法についての提案を、以下に示します。

1. **rah** がハングしている (または非常に長時間かかっている)。

   この問題は、下記が原因とみられます。

   - 出力をバッファーに入れる必要があると **rah** が判断したが、 RAHCHECKBUF=no が設定されなかった。このため、**rah** はコマンドを実行する前にコマンドをすべてのマシンに送って、バッファー・ディレクトリーの存在をチェックし、存在しなければバッファー・ディレクトリーを作成します。
   - コマンド送信先のマシンのうち、いくつかが応答していない。 **rsh** コマンドは最終的にタイムアウトになりますが、タイムアウトのインターバルは極めて長く、通常は約 60 秒です。

2. 次のようなメッセージを受け取った。
   - ログインの誤り
   - 許可が否定された

   いずれかのマシンの .hosts ファイル内で、 **rah** を実行する ID が正しく定義されていません。あるいは、 **rah** を実行する ID の .rhosts ファイル内で、いずれかのマシンが正しく定義されていません。

3. バックグラウンドの rshells を使用して並列にコマンドを実行している場合、コマンドが実行されてマシンで予期された経過時間内に完了するが、 **rah** がそれを検知してシェル・プロンプトを準備するのに時間がかかる。

   rah を実行している ID の .rhosts ファイル内で、いずれかのマシンが正しく定義されていません。

4. **rah** がシェル・コマンド行からは正しく実行されるものの、たとえば、次のように rsh を使ってリモートで **rah** を実行すると、

   ```
   rsh somewher -l $USER db2_kill
   ```

   rah が完了しない。

   これは正常です。 **rah** は、自らが終了した後にも実行を続けるバックグラウンドのモニター・プロセスを開始します。これらのプロセスは、通常、実行したコマンドに関連するすべてのプロセスが終了するまで続行します。 **db2_kill** の場合は、これは、すべてのデータベース・マネージャーの終了を意味します。関連するコマンドが **rahwaitfor** および kill <process_id> であるようなプロセスを探すことによって、モニター・プロセスを終了することができます。シグナル番号を指定してはなりません。代わりに、デフォルトの (15) にしておきます。

5. 同じ $RAHUSER の下で複数の **rah** コマンドが発行されると、 **rah** からの出力が正しく表示されない。または、**rah** は $RAHBUFNAME が存在しないと誤って報告する。

   これは、複数の **rah** が同時に実行して、出力のバッファリング用として同じバッファー・ファイル ($RAHBUFDIR/$RAHBUFNAME など) を使用しようとする

ためです。このような問題を避けるために、同時に実行されるそれぞれの **rah** コマンドごとに、別の $RAHBUFNAME を使用してください。たとえば、以下の ksh では、

```
export RAHBUFNAME=rahout
rah ";$command_1" &
export RAHBUFNAME=rah2out
rah ";$command_2" &
```

のようにするか、または次のようにして、一意の名前をシェルに自動的に選択させるようにします。

```
RAHBUFNAME=rahout.$$ db2_all "....."
```

どの方法を使用する場合も、ディスク・スペースに制約があるならば、いずれかの時点でバッファー・ファイルを確実に終結処理する必要があります。 **rah** は実行の終わりにバッファー・ファイルを消去しませんが、次に同じバッファー・ファイルを指定した場合、既存のファイルを消去して再使用します。

6. 次のように入力して、

```
rah '"print from ()'
```

以下のメッセージを受け取った。

```
ksh: syntax error at line 1 : (' unexpected
```

() および ## の置換の前提条件は次のとおりです。

- **rah** ではなく、 **db2_all** を使用する。
- RAHOSTFILE を設定するか、またはデフォルトである /sqllib/db2nodes.cfg ファイルにすることによって、 RAHOSTFILE が使用されていることを確認する。このような前提条件がない場合は、**rah** は、 () と ## を現状のままにします。コマンド **print from ()** は有効でないので、エラーになります。

コマンドを並列実行している場合、パフォーマンス改善のヒントとして、 & によって提供される機能が本当に必要でない限り、 |& ではなく | を、 ||& ; ではなく || を使用してください。 & を指定すると、**rsh** コマンドが必要になり、パフォーマンスが低下するためです。

関連資料:

- 389 ページの『rah コマンドの制御』

付録 E. Windows Management Instrumentation (WMI) サポートの使用

Windows Management Instrumentation (WMI) の紹介

管理インフラストラクチャーの規格を制定し、各種のハードウェアおよびソフトウェア管理システムの情報を結合するための方法を提供する、産業イニシアチブが存在します。このイニシアチブは、Web-Based Enterprise Management (WBEM) と呼ばれています。 WBEM は、Desktop Management Task Force (DMTF) 主導の業界標準である Common Information Model (CIM) スキーマを基にしています。

Microsoft® Windows® Management Instrumentation (WMI) は、サポートされる Windows プラットフォーム用に WBEM イニシアチブを実現したものです。 WMI は、Windows エンタープライズ・ネットワークで役立ちます。これにより、エンタープライズ・ネットワーク・コンポーネントの保守と管理費用が削減されます。 WMI は以下を提供します。

- Windows の操作、構成、および状況の一貫性のあるモデル。
- 管理情報にアクセスできるようにする COM API。
- 他の Windows 管理サービスと協働する機能。
- 柔軟で拡張可能なアーキテクチャー。これにより、ベンダーは、新しい装置、アプリケーション、その他の機能強化をサポートするための他の WMI プロバイダーを作成できます。
- 情報の詳細な照会を作成するための WMI Query Language (WQL)。
- 管理アプリケーション開発者が Visual Basic または Windows Scripting Host (WSH) スクリプトを作成するための API。

WMI アーキテクチャーには、以下の 2 つの部分があります。

1. 管理インフラストラクチャー。これには、CIM Object Manager (CIMOM) と、CIMOM オブジェクト・リポジトリーと呼ばれる管理データ用の中央ストレージ域が含まれています。 CIMOM を使用すると、アプリケーションは一様な方法で管理データにアクセスできるようになります。
2. WMI プロバイダー。 WMI プロバイダーは、CIMOM と管理下のオブジェクトの中間物です。 WMI API を使用することにより、WMI プロバイダーは、管理下のオブジェクトのデータを CIMOM に提供し、管理アプリケーションの代わりに要求を処理し、イベント通知を生成します。

Windows Management Instrumentation (WMI) プロバイダーは、管理下のオブジェクトと CIM Object Manager (CIMOM) の仲介機能としての役割を果たす、標準の COM または DCOM サーバーです。 CIMOM が、CIMOM オブジェクト・リポジトリーからは使用できないデータ (またはイベント) に対する要求を管理アプリケーションから受け取ると、 CIMOM はその要求を WMI プロバイダーに転送します。 WMI プロバイダーは、管理対象オブジェクトに対して、それぞれのドメインに固有のデータとイベント通知を提供します。

関連概念:

- 396 ページの『DB2 Universal Database と Windows Management Instrumentation の統合』

DB2 Universal Database と Windows Management Instrumentation の統合

Windows® Management Instrumentation (WMI) は、 DB2® パフォーマンス・カウンターを使って、また組み込み PerfMon プロバイダーを使用してスナップショット・モニターにアクセスできます。

WMI は、組み込みレジストリー・プロバイダーを使用して、 DB2 プロファイル・レジストリー変数にアクセスできます。

WMI Software Development Kit (WMI SDK) には、以下の複数の組み込みプロバイダーが含まれています。

- PerfMon プロバイダー
- レジストリー・イベント・プロバイダー
- レジストリー・プロバイダー
- Windows NT® イベント・ログ・プロバイダー
- Win32 プロバイダー
- WDM プロバイダー

WMI は、組み込み Windows NT イベント・ログ・プロバイダーを使用して、イベント・ログ内の DB2 エラーにアクセスできます。

DB2 Universal Database™ (UDB) には、以下の管理下のオブジェクトにアクセスするための、 DB2 WMI 管理プロバイダーとサンプルの WMI スクリプト・ファイルがあります。

1. 区画に分割されたインスタンスを含むデータベース・サーバーのインスタンス。以下の操作を実行できます。
   - インスタンスの列挙
   - データベース・マネージャー・パラメーターの構成
   - DB2 サーバー・サービスの開始/停止/状況照会
   - 通信のセットアップまたは確立
2. データベース。以下の操作を実行できます。
   - データベースの列挙
   - データベース・パラメーターの構成
   - データベースの作成/ドロップ
   - データベースのバックアップ/リストア/ロールフォワード

WMI アプリケーションを実行する前に、 DB2 WMI プロバイダーをシステムに登録する必要があります。以下のコマンドを入力して登録を行います。

- mofcomp %DB2PATH%¥bin¥db2wmi.mof

このコマンドは、DB2 WMI スキーマの定義をシステムにロードします。

- regsvr %DB2PATH%¥bin¥db2wmi.dll

  このコマンドは、DB2 WMI プロバイダー COM DLL を Windows に登録します。

両方のコマンドで、 %DB2PATH% は DB2 のインストール先のパスです。また、db2wmi.mof は DB2 WMI スキーマ定義が入っている .MOF ファイルです。

WMI インフラストラクチャーを統合することには、以下のような複数の利点があります。

1. WMI 提供のツールを使用して、 Windows ベースの環境で DB2 サーバーを管理するためのスクリプトを簡単に作成できます。インスタンスのリスト、データベースの作成とドロップ、構成パラメーターの更新などの単純なタスクを実行するための、サンプルの Visual Basic (VBS) スクリプトが提供されています。サンプル・スクリプトは、DB2 Application Development for Windows 製品に組み込まれています。
2. WMI を使用して多くのタスクを実行する強力な管理アプリケーションを作成できます。タスクには以下のものがあります。
   - システム情報の表示
   - DB2 パフォーマンスのモニター
   - DB2 システム・リソース使用量のモニター

   このタイプの管理アプリケーションを使って、システム・イベントと DB2 イベントの両方をモニターすることにより、データベースをよりよく管理できます。
3. 既存の COM および Visual Basic プログラミングの知識とスキルを活用できます。 COM または Visual Basic インターフェースにより、プログラマーは、エンタープライズ管理アプリケーションの開発時間を短縮できます。

関連概念:

- 395 ページの『Windows Management Instrumentation (WMI) の紹介』

付録 F. Windows NT セキュリティーの使用

DB2 for Windows NT および Windows NT セキュリティーの紹介

Windows® NT ドメインは、特定の名前およびユニークな名前で参照されるクライアント・コンピューターおよびサーバー・コンピューターの集合であり、 Security Access Manager (SAM) と呼ばれる単一のユーザー・アカウント・データベースを共用します。ドメイン内のコンピューターのうちの 1 つが、ドメイン・コントローラーです。ドメイン・コントローラーは、ユーザー・ドメインの対話のすべての面を管理します。ドメイン・コントローラーは、ドメイン・アカウントにログオンするユーザーを認証するために、ドメイン・ユーザー・アカウント・データベース内の情報を使用します。ドメインごとに、1 つのドメイン・コントローラーが 1 次ドメイン・コントローラー (PDC) になります。ドメイン内には、1 次ドメイン・コントローラーが存在しない場合、または 1 次ドメイン・コントローラーが使用不可の場合に、ユーザー・アカウントを認証するバックアップ・ドメイン・コントローラー (BDC) が存在する場合があります。バックアップ・ドメイン・コントローラーは、PDC のマスター・コピーと定期的に同期化される SAM データベースのコピーを保持しています。

ユーザー・アカウント、ユーザー ID、およびパスワードは、 1 次ドメイン・コントローラーに定義するだけで、ドメイン・リソースにアクセスできるようになります。

Windows NT® サーバーのインストール時のセットアップ手順の中で、以下を作成するように選択できます。

- 1 次ドメイン・コントローラー (新しいドメイン内)
- バックアップ・ドメイン・コントローラー (既知のドメイン内)
- スタンドアロン・サーバー (既知のドメイン内)

新しいドメイン内で「コントローラー」を選択すると、そのサーバーは 1 次ドメイン・コントローラーになります。

ユーザーはローカル・マシンにログオンすることができます。あるいは、 Windows NT ドメイン中にマシンをインストールしているならば、ユーザーはそのドメインにログオンできます。 DB2® (Windows NT 版) はこれら両方の機能をサポートします。ユーザーを認証するために、 DB2 は最初にローカル・マシンのリスト、次に現在のドメインのドメイン・コントローラー、最後にドメイン・コントローラーを認識する承認されたドメインをチェックします。

この動作の仕方を説明するために、DB2 インスタンスがサーバー認証を必要とすると仮定します。構成は、以下のとおりです。

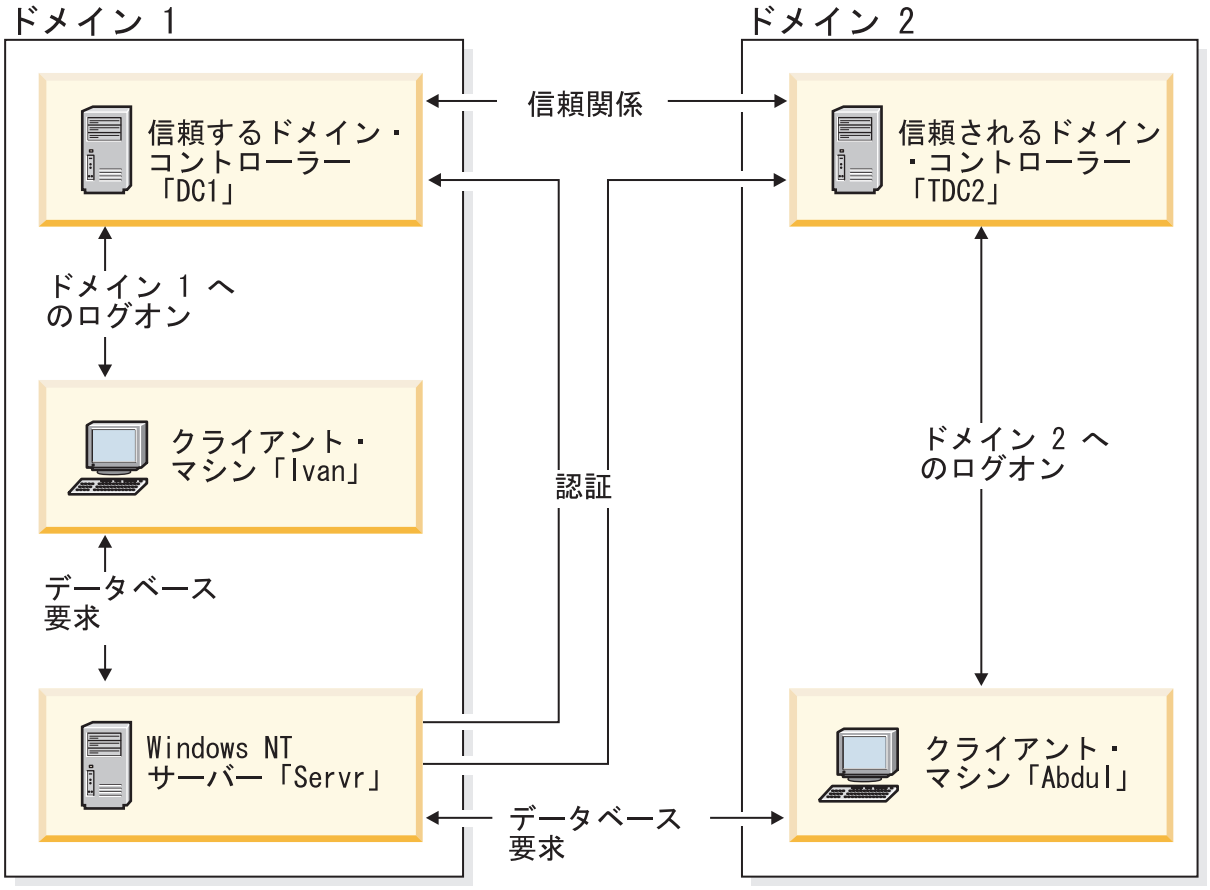


図 6. Windows NT のドメインを使用した認証

各マシンには、クライアント・マシンが Windows 9x を実行していない限り、セキュリティー・データベース (セキュリティー・アクセス管理 (SAM)) があります。 Windows 9x マシンには、SAM データベースはありません。 DC1 はドメイン・コントローラーで、そのクライアント・マシンの Ivan、 DB2 (Windows NT 版) のサーバーの Servr が登録されています。 TDC2 は DC1 に承認されたドメインで、クライアント・マシンの Abdul は TDC2 のドメインのメンバーです。

関連概念:

- 404 ページの『Windows NT でのグループ認証およびユーザー認証』

関連タスク:

- 403 ページの『DB2 UDB でのバックアップ・ドメイン・コントローラーの使用』
- 406 ページの『DB2 のバックアップ・ドメイン・コントローラーへのインストール』
- 407 ページの『DB2 for Windows NT 認証でのグループおよびドメイン・セキュリティーの使用』

DB2 for Windows NT でサーバー認証を使用するシナリオ

1. Abdul は、TDC2 ドメインにログオンします (つまり、 TDC2 SAM データベースに認識されています)。

2. 次に Abdul は、次のように入力して SRV3 上に常駐するためにカタログされた DB2 データベースに接続します。

```
db2 connect to remotedb user Abdul using fredpw
```

3. SRV3 は、Abdul が認識されている場所を判別します。この情報を検出するのに使われる API は、最初にローカル・マシン (SRV3)、次にドメイン・コントローラー (DC1) を検索し、最後に承認されたドメインを検索しようとします。ユーザー名 Abdul が TDC2 上で検出されます。この検索順序には、ユーザーとグループに単一ネーム・スペースが必要です。
4. 次に SRV3 は、次のように実行します。
   a. TDC2 を使ってユーザー名とパスワードの妥当性を検査します。
   b. TDC2 に照会することによって、Abdul が管理者かどうかを検出します。
   c. TDC2 に照会することによって、 Abdul のグループすべてを列挙します。

関連概念:

- 399 ページの『DB2 for Windows NT および Windows NT セキュリティーの紹介』

DB2 for Windows NT でクライアント認証および Windows NT クライアント・マシンを使用するシナリオ

1. 管理者の Dale は、SRV3 にログオンし、クライアントに対するデータベース・インスタンスの認証を変更します。

```
db2 update dbm cfg using authentication client
db2stop
db2start
```

2. Windows クライアント・マシンで、Ivan は DC1 ドメインにログオンします (つまり、 DC1 SAM データベースに認識されています)。
3. 次に Ivan は、次のように入力して SRV3 上に常駐するためにカタログされた DB2 データベースに接続します。

```
DB2 CONNECT to remotedb user Ivan using johnpw
```

4. Ivan のマシンは、ユーザー名とパスワードの妥当性を検査します。この情報を検出するのに使われる API は、最初にローカル・マシン (Ivan)、次にドメイン・コントローラー (DC1) を検索し、最後に承認されたドメインを検索しようとします。ユーザー名 Ivan が DC1 上で検出されます。
5. 次に Ivan のマシンは、DC1 を使ってユーザー名とパスワードの妥当性を検査します。
6. 次に SRV3 は、次のように実行します。
   a. Ivan が認識された場所を判別します。
   b. DC1 に照会することによって、Ivan は管理者かどうかを検出します。
   c. DC1 に照会することによって、Ivan のすべてのグループを列挙します。

注: DB2 データベースに接続してみる前に、必ず DB2 セキュリティー・サービスを始動してください。セキュリティー・サービスは、Windows のインストールの一部としてインストールされます。次いで DB2 がインストールされ、Windows NT サービスとして「登録」され、デフォルトでは、自動的に開始されます。 DB2 セキュリティー・サービスを始動するには、 `NET START DB2NTSECSERVER` コマンドを入力してください。

関連概念:

- 399 ページの『DB2 for Windows NT および Windows NT セキュリティーの紹介』

DB2 for Windows NT でクライアント認証および Windows 9x クライアント・マシンを使用するシナリオ

1. 管理者の Dale は、SRV3 にログオンし、クライアントに対するデータベース・インスタンスの認証を変更します。

   ```
   db2 update dbm cfg using authentication client
   db2stop
   db2start
   ```

2. Windows 9x マシンで、Ivan は DC1 ドメインにログオンします (つまり、 DC1 SAM データベースに認識されています)。
3. 次に Ivan は、次のように入力して SRV3 上に常駐するためにカタログされた DB2 データベースに接続します。

   ```
   db2 connect to remotedb user Ivan using johnpw
   ```

4. Ivan の Windows 9x マシンはユーザー名とパスワードの妥当性を検査することはできません。したがって、ユーザー名とパスワードは有効と見なされます。
5. 次に SRV3 は、次のように実行します。
   a. Ivan が認識された場所を判別します。
   b. DC1 に照会することによって、Ivan は管理者かどうかを検出します。
   c. DC1 に照会することによって、Ivan のすべてのグループを列挙します。

注: Windows 9x クライアントは、与えられたユーザー名とパスワードの妥当性を検査できないため、 Windows 9x でのクライアント認証は本質的に不確実です。Windows 9x マシンが Windows NT セキュリティー・プロバイダーにアクセスした場合、妥当性検査されたパススルー・ログオンのために Windows 9x システムを構成することによって、セキュリティーのいくつかの基準を課することができます。この方法で Windows 9x システムを構成する方法の詳細については、 Microsoft 社の Windows 9x の資料を参照してください。

関連概念:

- 399 ページの『DB2 for Windows NT および Windows NT セキュリティーの紹介』
- 402 ページの『Windows でのグローバル・グループのサポート』

Windows でのグローバル・グループのサポート

さらに、DB2® Universal Database (DB2 UDB) はグローバル・グループもサポートします。グローバル・グループを使用するために、ローカル・グループ内にグローバル・グループを組み込む必要があります。あるユーザーがメンバーとなっているグループを DB2 UDB がすべて列挙するとき、そのユーザーが間接的にメンバーになっているローカル・グループもまたリストされます (そのグループは、 1 つまたは複数のローカル・グループのメンバーになっているグローバル・グループ内にあるため)。

グローバル・グループは、以下の 2 つの状態で使用されます。

- ローカル・グループに組み込まれた状態。ローカル・グループに許可を付与する必要があります。
- ドメイン・コントローラーに組み込まれた状態。グローバル・グループに許可を付与する必要があります。

関連概念:

- 404 ページの『Windows NT でのグループ認証およびユーザー認証』

DB2 UDB でのバックアップ・ドメイン・コントローラーの使用

手順:

また、DB2 Universal Database™ (DB2 UDB) 用に使用しているサーバーがバックアップ・ドメイン・コントローラーとしても動作する場合、 DB2 を構成してバックアップ・ドメイン・コントローラーを使用すれば、 DB2 UDB のパフォーマンスを向上させ、ネットワーク・トラフィックを削減することができます。

DB2DMNBCKCTLR レジストリー変数を設定することによって、 DB2 UDB にバックアップ・ドメイン・コントローラーを指定します。

DB2 UDB サーバーがバックアップ・ドメイン・コントローラーになっている場合、そのドメイン・ネームが分かれば、次を使用します。

```
db2dmnbckctlr=<domain_name>
```

この場合、domain_name は大文字でなければなりません。

ローカル・マシンがバックアップ・ドメイン・コントローラーになっている場合、DB2 UDB がそのドメインを判別していれば、以下を使用します。

```
DB2DMNBCKCTLR=?
```

注: デフォルトでは、DB2 UDB はバックアップ・ドメイン・コントローラーを使用しません。バックアップ・ドメイン・コントローラーは 1 次ドメイン・コントローラーと同期しないことがあり、機密漏れを生じることがあるためです。1 次ドメイン・コントローラーのセキュリティー・データベースが更新されても、その変更内容がバックアップ・ドメイン・コントローラーに伝搬していない場合に、ドメイン・コントローラーが同期しなくなることがあります。この事態は、ネットワーク待ち時間が生じた場合や、コンピューターのブラウザー・サービスが作動可能でない場合に起こることがあります。

関連タスク:

- 406 ページの『DB2 のバックアップ・ドメイン・コントローラーへのインストール』

DB2 for Windows NT での DB2 ユーザー認証

Windows NT ユーザーの場合、オペレーティング・システムの認証の方法が原因で、ユーザーの認証に問題の生じることがあります。この節では、 DB2 (Windows NT 版) でのユーザー認証に関する考慮事項について取り上げます。

- 『DB2 for Windows NT でのユーザー名およびグループ名に関する制約事項』
- 406 ページの『DB2 for Windows NT セキュリティー・サービス』
- 406 ページの『DB2 のバックアップ・ドメイン・コントローラーへのインストール』
- 407 ページの『DB2 for Windows NT 認証でのグループおよびドメイン・セキュリティーの使用』

DB2 for Windows NT でのユーザー名およびグループ名に関する制約事項

Windows NT 環境では、次のような制約があります。

- DB2 Universal Database™ (DB2 UDB) の内部では、ユーザー名およびグループ名は 30 文字以下に制限されています。
- Windows NT 環境では、ユーザー名に大文字小文字の区別はありません。ただし、パスワードには大文字小文字の区別があります。
- ユーザー名やグループ名には大文字と小文字の両方を含めることができます。ただし、DB2 UDB 内で使用されるとき、通常は大文字に変換されます。たとえば、データベースに接続してから表 schema1.table1 を作成した場合、この表はデータベース内に SCHEMA1.TABLE1 として保管されます。 (小文字のオブジェクト名を使用したい場合は、コマンド行プロセッサーからコマンドを発行するときにオブジェクト名を引用符で囲むか、あるいはサード・パーティーの ODBC フロントエンド・ツールを使用します。)
- ユーザーが属すことのできるグループの数は 64 までです。
- DB2 UDB は単一のネーム・スペースをサポートします。つまり、トラステッド・ドメイン環境で実行する場合は、同じ名前のユーザー・アカウントを複数のドメインに置いたり、サーバー・マシンのローカル SAM や別のドメインに置いたりすることはできません。

関連概念:

- 404 ページの『Windows NT でのグループ認証およびユーザー認証』
- 405 ページの『Windows NT でのドメイン間の信頼関係』

Windows NT でのグループ認証およびユーザー認証

ユーザーは、「ユーザ マネージャ」という Windows® NT 管理ツールを使用して、Windows® NT 上に定義されます。

他のアカウント (メンバーとも呼ばれる) が入っているアカウントは、グループです。グループを使用すれば、Windows NT 管理者は、権利および許可をグループ内のユーザーに一度に付与できるようになり、各ユーザーを個別に保守する必要がなくなります。グループは、ユーザー・アカウントと同様、 Security Access Manager (SAM) データベース内で定義され保守されます。

グループには次の 2 つの種類があります。

- ローカル・グループ。ローカル・グループには、ローカル・アカウント・データベース内で作成されたユーザー・アカウントを入れることができます。ローカ

ル・グループがドメインの一部であるマシン上に存在する場合は、ローカル・グループには、 Windows NT ドメインのドメイン・アカウントおよびグループを入れることもできます。ローカル・グループをワークステーション上に作成する場合は、ローカル・グループはそのワークステーション固有です。

- グローバル・グループ。グローバル・グループは、ドメイン・コントローラー上にのみ存在し、ドメインの SAM データベースのユーザー・アカウントが含まれています。つまり、グローバル・グループには、グローバル・グループが作成されたドメインのユーザー・アカウントだけを入れることができます。他のグループをメンバーとして入れることはできません。グローバル・グループは、グローバル・グループが所属するドメインのサーバーおよびワークステーションと、トラスティング・ドメインで使用できます。

関連概念:

- 405 ページの『Windows NT でのドメイン間の信頼関係』
- 402 ページの『Windows でのグローバル・グループのサポート』

関連タスク:

- 407 ページの『DB2 for Windows NT 認証でのグループおよびドメイン・セキュリティーの使用』

関連資料:

- 404 ページの『DB2 for Windows NT でのユーザー名およびグループ名に関する制約事項』

Windows NT でのドメイン間の信頼関係

信頼関係は、2 つのドメイン間の管理および通信リンクです。 2 つのドメイン間に信頼関係があれば、ユーザー・アカウントおよびグローバル・グループを、アカウントが定義されたドメインではないドメインで使用できるようになります。アカウント情報は、トラステッド・ドメイン内に認証されずに存在しているユーザー・アカウントおよびグローバル・グループの権利や許可を検証するために共用されます。信頼関係は、2 つ以上のドメインを単一の管理単位にまとめることにより、ユーザーの管理を単純化します。

信頼関係には次の 2 つのドメインがあります。

- トラスティング・ドメイン。このドメインは、ユーザーを認証してもらうために別のドメインを信頼しています。
- トラステッド・ドメイン。このドメインは、別のドメインに代わってユーザーを認証します。

信頼関係は他動的なものではありません。つまり、ドメイン間で双方向の信頼関係を明示的に確立する必要があります。たとえば、トラスティング・ドメインは、必ずしもトラステッド・ドメインであるとは限りません。

関連概念:

- 404 ページの『Windows NT でのグループ認証およびユーザー認証』
- 402 ページの『Windows でのグローバル・グループのサポート』

関連資料:

- 404 ページの『DB2 for Windows NT でのユーザー名およびグループ名に関する制約事項』

DB2 for Windows NT セキュリティー・サービス

DB2® Universal Database (DB2 UDB) では、ユーザー名とパスワードの認証が DB2 システム・コントローラーに統合されています。セキュリティー・サービスが必要になるのは、認証 CLIENT 用に構成されたサーバーにクライアントが接続するときだけです。

関連概念:

- 399 ページの『DB2 for Windows NT および Windows NT セキュリティーの紹介』

DB2 のバックアップ・ドメイン・コントローラーへのインストール

手順:

Windows NT 4.0 環境では、プライマリー・コントローラーとバックアップ・コントローラーのどちらでもユーザーの認証を実行できます。この機能は、どのサイトにも 1 つの中央 1 次ドメイン・コントローラー (PDC) と、 1 つまたは複数のバックアップ・ドメイン・コントローラー (BDC) とが配置されているような大規模な分散 LAN では非常に重要です。ユーザーは認証のために 1 次ドメイン・コントローラーを呼び出さなくても、現在のサイトにあるバックアップ・ドメイン・コントローラーで認証を実行できます。

この場合、バックアップ・ドメイン・コントローラーを設けることの利点は、ユーザーの認証を短時間で行えることと、 BDC がない場合よりも LAN が混雑しないで済むことです。

以下の条件にあてはまる場合、認証は BDC で行うことができます。

- DB2 (Windows NT 版) サーバーがバックアップ・ドメイン・コントローラーにインストールされている場合。
- DB2DMNBCKCTLR プロファイル・レジストリー変数が正しく設定されている場合。

DB2DMNBCKCTLR プロファイル・レジストリー変数が設定されていない場合、あるいはブランクに設定されている場合、 DB2 (Windows NT 版) は認証を 1 次ドメイン・コントローラーで実行します。

DB2DMNBCKCTLR に有効な宣言設定は ″?″ またはドメイン・ネームだけです。

DB2DMNBCKCTLR プロファイル・レジストリー変数が疑問符に設定されていて (DB2DMNBCKCTLR=?)、かつ以下の条件にあてはまる場合、 DB2 (Windows NT 版) は認証をバックアップ・ドメイン・コントローラーで実行します。

- cachedPrimaryDomain レジストリー値が、このマシンが属しているドメインの名前に設定されている。 (この設定を調べるには、

「HKEY_LOCAL_MACHINE」-> 「ソフトウェア (Software)」-> 「Microsoft」-> 「Windows NT」-> 「現行バージョン (Current Version)」-> 「WinLogon」を選択します。)

- サーバー・マネージャーによって、バックアップ・ドメイン・コントローラーがアクティブかつ使用可能であることが示されている。 (つまり、このマシンを表すアイコンがグレー表示にはなっていない。)
- DB2 Windows NT サーバーのレジストリーによって、そのシステムが指定されたドメイン上のバックアップ・ドメイン・コントローラーであることが示されている。

通常の状態であれば DB2DMNBCKCTLR=? 設定はたいてい正常に機能しますが、あらゆる環境で正常に機能することが保証されているわけではありません。ドメイン上のサーバーについて提供される情報は動的であるため、コンピューター・ブラウザーを実行して、この情報を常に正確かつ最新のものに保つ必要があります。大規模な LAN ではコンピューター・ブラウザーを実行できないことがあるため、サーバー・マネージャーの情報が最新のものではなくなる場合もあります。この場合、 DB2 (Windows NT 版) に認証をバックアップ・ドメイン・コントローラーで実行させる別の方法があります。それは、DB2DMNBCKCTLR=*xxx* を設定することです (*xxx* は DB2 サーバーの Windows NT ドメイン・ネーム)。このように設定されている場合、以下の条件を満たしていれば、認証がバックアップ・ドメイン・コントローラーで行われます。

- cachedPrimaryDomain レジストリー値が、このマシンが属しているドメインの名前に設定されている。 (この設定を調べるには、**「HKEY_LOCAL_MACHINE」-> 「ソフトウェア (Software)」-> 「Microsoft」-> 「Windows NT」-> 「現行バージョン (Current Version)」-> 「WinLogon」**を選択します。)
- マシンが、指定されたドメイン用のバックアップ・ドメイン・コントローラーとして構成されている。 (マシンがバックアップ・ドメイン・コントローラーとしてセットアップされていても、他のドメインのためのものである場合、この設定はエラーになります。)

関連タスク:

- 403 ページの『DB2 UDB でのバックアップ・ドメイン・コントローラーの使用』

DB2 for Windows NT 認証でのグループおよびドメイン・セキュリティーの使用

手順:

DB2 Universal Database™ (DB2 UDB) では、特権を付与するときまたは権限レベルを定義する時に、ローカル・グループかグローバル・グループを指定できます。ユーザーがグループのメンバーであると判断されるのは、そのユーザーのアカウントが、ローカルまたはグローバル・グループ内で明示的に定義されている場合、またはローカル・グループのメンバーになるように定義されているグローバル・グループのメンバーになることによって暗黙的に定義されている場合です。

DB2 (Windows NT 版) は、以下のタイプのグループをサポートします。

- ローカル・グループ
- グローバル・グループ
- ローカル・グループのメンバーとしてのグローバル・グループ

  DB2 (Windows NT 版) は、ユーザーの情報が含まれているデータベースを使用して、そのユーザーがメンバーとなっているローカル・グループとグローバル・グループを列挙します。 DB2 UDB は、ユーザー・アカウントがどこにあるかに関係なく、 DB2 UDB がインストールされているローカル Windows NT サーバーでグループが列挙されるようにオーバーライドを強制します。このオーバーライドを実行するには、以下のコマンドを使用します。

  – グローバル設定の場合:

    ```
    db2set -g DB2_GRP_LOOKUP=local
    ```

  – インスタンス設定の場合:

    ```
    db2set -i <instance name> DB2_GRP_LOOKUP=local
    ```

    このコマンドの発行後、変更を有効にするには、 DB2 UDB インスタンスを停止して開始する必要があります。次に、ローカル・グループを作成し、ドメイン・アカウントまたはグローバル・グループをそのローカル・グループに入れます。

  設定されているすべての DB2 プロファイル・レジストリー変数を表示するには、次のように入力します。

  ```
  db2set -all
  ```

DB2_GRP_LOOKUP プロファイル・レジストリー変数が local に設定されている場合、 DB2 UDB はローカル・マシンでのみユーザーを探そうとします。そのユーザーがローカル・マシンで見つからなかった場合や、そのユーザーがローカルまたはグローバル・グループのメンバーとして定義されていない場合、認証は失敗します。 DB2 は、同じドメインの他のマシンやドメイン・コントローラーからユーザーを探そうとは**しません**。

DB2_GRP_LOOKUP プロファイル・レジストリー変数が設定されていない場合、以下が行われます。

1. DB2 UDB は最初に同じマシン上でユーザーを探そうとします。
2. ユーザー名がローカルで定義されている場合、そのユーザーの認証はローカルで行われます。
3. ユーザーがローカルで見つからなかった場合、 DB2 UDB は同じドメインの中からユーザー名を探そうとし、それでも見つからない場合は、トラステッド・ドメインから探そうとします。

リソース・ドメイン内で 1 次ドメイン・コントローラーまたは バックアップ・ドメイン・コントローラーであるマシン上で DB2 UDB が実行されている場合は、任意のドメイン・コントローラーを任意のトラステッド・ドメインに置くことができます。トラステッド・ドメイン内のバックアップ・ドメイン・コントローラーのドメインの名前は、ドメイン・コントローラーでなければ知ることができないためです。

DB2 UDB がドメイン・コントローラー上で実行されていない場合は、以下を発行する必要があります。

```
db2set -g DB2_GRP_LOOKUP=DOMAIN
```

このコマンドは、DB2 UDB が属するドメイン内のドメイン・コントローラーを使用して、アカウント・ドメイン内のドメイン・コントローラーの名前を検索するように DB2 に通知します。つまり、特定のユーザー・アカウントがドメイン x に定義されていることが DB2 UDB によって検出される場合は、 DB2 UDB はドメイン x のドメイン・コントローラーを探そうとするのではなく、その要求を DB2 UDB が属するドメイン内のドメイン・コントローラーに送信します。アカウント・ドメイン内のドメイン・コントローラーの名前が検出され、 DB2 UDB が実行されているマシンに戻されます。この方法には、以下の 2 つの利点があります。

1. 1 次ドメイン・コントローラーが使用できない場合に、バックアップ・ドメイン・コントローラーが検出される。
2. 1 次ドメイン・コントローラーが地理的に離れている場合は、近くにあるバックアップ・ドメイン・コントローラーが検出される。

関連概念:

- 404 ページの『Windows NT でのグループ認証およびユーザー認証』

順序付けドメイン・リストを使用した認証

1 つのトラステッド・ドメイン・フォレスト内では、ユーザー ID が複数回にわたって定義される場合があります。トラステッド・ドメイン・フォレストとは、ネットワークを介して互いに関連している複数のドメインからなる集合です。 1 つのドメインのユーザーが、別のドメイン内の別のユーザーと同じユーザー ID を持つ可能性があります。そのような場合、以下の操作が困難になります。

- 同じユーザー ID を持つ複数のユーザーをそれぞれ別のドメインで認証する。
- グループに基づいて特権を付与または取り消すための、グループ・ルックアップ。
- パスワードの検証。
- ネットワーク・トラフィックの制御。

手順:

同じユーザー ID を持つ複数のユーザーがドメイン・フォレスト内でアクセスする場合の問題を防ぐには、 **db2set** およびレジストリー変数 DB2DOMAINLIST を使って定義される、順序付けドメイン・リストを使用する必要があります。順序を設定するときには、リストに含める複数のドメインをコンマで区切ります。ユーザー認証時に複数のドメインを検索する順序を決定するときには、十分に考慮する必要があります。

ドメイン・リストの下の方にあるドメインに含まれるユーザー ID がアクセスのために認証されるには、それらを名前変更しなければなりません。

ドメイン・リストを介してアクセスを制御することができます。たとえば、ユーザーのドメインがリストに含まれない場合、そのユーザーは接続を許可されません。

注: DB2DOMAINLIST レジストリー変数が有効になるのは、データベース・マネージャー構成で CLIENT 認証が設定され、 Windows NT ドメイン環境の Windows NT デスクトップからのシングル・サインオンでこの認証が必要とされる場合のみです。

関連概念:

- 399 ページの『DB2 for Windows NT および Windows NT セキュリティーの紹介』

DB2 for Windows NT のドメイン・セキュリティー・サポート

以下の例は、 DB2 (Windows NT 版) がドメイン・セキュリティーをどのようにサポートするかを示しています。最初の例では、ユーザー名とローカル・グループが同じドメイン上にあるため、接続は機能します。 2 番目の例では、ユーザー名とローカルまたはグローバル・グループが異なるドメイン上にあるため、接続は機能しません。

接続が成功する例: 以下のシナリオでは、ユーザー名とローカルまたはグローバル・グループが同じドメイン上にあるため、接続は機能します。

必ずしもユーザー名とローカルまたはグローバル・グループを、データベース・サーバーが実行されているドメインに定義する必要はありません。しかし、ユーザー名とローカルまたはグローバル・グループを同じドメインに定義する必要はあります。

表 52. ドメイン・コントローラーを使用した接続が成功する場合

Domain1	Domain2
Domain2 との間に信頼関係が存在している。	• Domain1 との間に信頼関係が存在している。 • ローカルまたはグローバル・グループ grp2 が定義されている。 • ユーザー名 id2 が定義されている。 • ユーザー名 id2 が grp2 のメンバーとなっている。

DB2 サーバーがこのドメインで実行されている。以下の DB2 コマンドがこのサーバーから発行される。
`REVOKE CONNECT ON db FROM public`
`GRANT CONNECT ON db TO GROUP grp2`
`CONNECT TO db USER id2`	

ローカルまたはグローバル・ドメインがスキャンされるが、 id2 は見つからない。ドメイン・セキュリティーがスキャンされる。	
	ユーザー名 id2 がこのドメインで見つかる。 DB2 は、このユーザー名についての追加情報 (つまり、このユーザー名がグループ grp2 のメンバーであるということ) を入手する。
ユーザー名とローカルまたはグローバル・グループが同じドメイン上にあるため、接続は機能する。	

関連概念:

- 404 ページの『Windows NT でのグループ認証およびユーザー認証』

関連タスク:

- 407 ページの『DB2 for Windows NT 認証でのグループおよびドメイン・セキュリティーの使用』

付録 G. Windows パフォーマンス・モニターの使用

Windows パフォーマンス・モニターの紹介

DB2® Universal Database (DB2 UDB) for Windows® では、パフォーマンスをモニターする以下のツールを使用できます。

- **DB2 Performance Expert**

  DB2 Performance Expert for Multiplatforms バージョン 1.1 は、 DB2 UDB パフォーマンス関連情報に基づいて、自己管理やリソース調整の変更を統合、報告、分析、および推奨します。

- **DB2 UDB ヘルス・センター**

  ヘルス・センターの機能は、パフォーマンス関連情報を利用するさまざまな方法を提供します。これらの機能を、コントロール・センターのパフォーマンス・モニターの一部の機能の代わりに使用することもできます。

- **Windows パフォーマンス・モニター**

  Windows パフォーマンス・モニターを使用すると、データベースとシステム・パフォーマンスの両方をモニターでき、そのシステムに登録されている任意のパフォーマンス・データ提供元から情報を取り出すことができます。さらに Windows は、以下を含めたマシン操作のすべてについて、パフォーマンス情報データを提供します。

  – CPU 使用状況
  – メモリー使用率
  – ディスクの活動状況
  – ネットワークの活動状況

関連タスク:

- 413 ページの『Windows パフォーマンス・モニターへの DB2 の登録』
- 414 ページの『DB2 パフォーマンス情報へのリモート・アクセスを使用可能にする』
- 415 ページの『DB2 UDB と DB2 Connect のパフォーマンス値を表示する』
- 417 ページの『リモートの DB2 UDB パフォーマンス情報へのアクセス』
- 417 ページの『DB2 パフォーマンス値をリセットする』

関連資料:

- 416 ページの『Windows パフォーマンス・オブジェクト』

Windows パフォーマンス・モニターへの DB2 の登録

手順:

セットアップ・プログラムは、 DB2 を自動的に Windows パフォーマンス・モニターへ登録します。

Windows パフォーマンス・モニターを使用し、 DB2 Universal Database™ (DB2 UDB) および DB2 Connect のパフォーマンス情報にアクセスできるようにするには、 DB2 (Windows 版) のパフォーマンス・カウンター用の DLL を登録する必要があります。またここで登録しておけば、 Win32 パフォーマンス API を使用してパフォーマンス・データを入手する Windows アプリケーションが他にあれば、そのアプリケーションも使用できるようになります。

DB2 (Windows 版) パフォーマンス・カウンターの DLL (DB2Perf.DLL) を、Windows パフォーマンス・モニターにインストールして登録するには、次のように入力します。

```
db2perfi -i
```

DLL を登録するならば、レジストリーのサービス・オプションで、新しいキーを作成することもできます。 1 つの項目には DLL の名前が示され、カウンターをサポートします。他の 3 つの項目には、その DLL に備えられている機能の名前が示されます。それらの機能は、以下のとおりです。

- **オープン**

  処理中に、DLL がシステムによって初めてロードされるときに呼び出されます。
- **収集**

  DLL からのパフォーマンス情報を要求するときに呼び出されます。
- **クローズ**

  DLL をアンロードするときに呼び出されます。

関連資料:

- 「コマンド・リファレンス」の『db2perfi - パフォーマンス・カウンター登録ユーティリティー・コマンド』

DB2 パフォーマンス情報へのリモート・アクセスを使用可能にする

手順:

ご使用の DB2 (Windows 版) ワークステーションが、別の Windows マシンにネットワークで接続されている場合、この節で説明されている機能を使用できます。

別の DB2 (Windows 版) マシンから Windows パフォーマンス・オブジェクトを見るには、 DB2 Universal Database™ (DB2 UDB) に管理者のユーザー名とパスワードを登録しなければなりません。 (デフォルトの Windows パフォーマンス・モニターのユーザー名である **SYSTEM** は、 DB2 UDB 予約語なので使用できません。) 名前を登録するには、次のように入力します。

```
db2perfr -r username password
```

注: 使用する `username` は、 DB2 UDB 命名規則に適合しなければなりません。

ユーザー名とパスワードのデータは、レジストリー内のキーに置かれます。このときのセキュリティーは、管理者および SYSTEM アカウントだけがアクセスできるというものです。管理者のパスワードをレジストリーに格納する際のセキュリティー上の問題を防ぐため、データはエンコードされます。

注:

1. いったんユーザー名とパスワードの組み合わせを DB2 UDB に登録すれば、パフォーマンス・モニターのローカル・インスタンスであっても、そのユーザー名とパスワードを使って明示的にログオンします。つまり、DB2 UDB に登録されたユーザー名情報が一致しなければ、パフォーマンス・モニターのローカル・セッションには、 DB2 UDB のパフォーマンス情報が示されないことになります。
2. ユーザー名とパスワードの組み合わせは、 Windows のセキュリティー・データベースに格納されているユーザー名とパスワードと常に一致させる必要があります。 Windows セキュリティー・データベース内のユーザー名かパスワードを変更した場合、リモートのパフォーマンス・モニターに使うユーザー名とパスワードの組み合わせを再設定しなければなりません。
3. 登録するには、次のように入力します。

```
db2perfr -u <username> <password>
```

関連概念:

- 323 ページの『一般的な命名規則』

関連資料:

- 「コマンド・リファレンス」の『db2perfr - パフォーマンス・モニター登録ツール・コマンド』

DB2 UDB と DB2 Connect のパフォーマンス値を表示する

手順:

パフォーマンス・モニターを使って DB2 Universal Database™ (DB2 UDB) および DB2 Connect のパフォーマンス値を表示するには、**「追加先 (Add to)」**ボックスから、表示させる値を示すパフォーマンス・カウンターを選択します。このボックスには、パフォーマンス・データを提示するパフォーマンス・オブジェクトのリストが示されます。提供されているカウンターのリストを見るには、特定のオブジェクトを選択してください。

1 つのパフォーマンス・オブジェクトに、複数のインスタンスが存在することもあります。たとえば、LogicalDisk オブジェクトには、″% Disk Read Time″ や ″Disk Bytes/sec″ などのカウンターが備えられています。さらに、マシン内の論理ドライブ (″C:″ や ″D:″ など) ごとに、 1 つのインスタンスがあります。

関連概念:

- 413 ページの『Windows パフォーマンス・モニターの紹介』

関連資料:

- 416 ページの『Windows パフォーマンス・オブジェクト』

Windows パフォーマンス・オブジェクト

Windows には、以下のパフォーマンス・オブジェクトがあります。

- **DB2 データベース・マネージャー**

  このオブジェクトは、1 つの Windows インスタンスのための、一般的な情報を提供します。モニターされる DB2 Universal Database™ (DB2 UDB) インスタンスは、オブジェクト・インスタンスとして表されます。

  実用上ならびにパフォーマンス上の理由のため、パフォーマンス情報は、一度に 1 つの DB2 UDB インスタンスだけから入手されます。パフォーマンス・モニターが示す DB2 UDB インスタンスは、パフォーマンス・モニターの処理では、db2instance レジストリー変数によって管理されます。同時に複数の DB2 UDB インスタンスを実行していて、 2 つ以上のパフォーマンス情報を確認したい場合、パフォーマンス・モニターのセッションを個別に開始する必要があります。このとき、db2instance には、モニターする DB2 UDB インスタンスごとに対応する値を設定します。

  パーティション・データベース・システムを実行している場合は、一度に 1 つのデータベース・パーティション・サーバー (ノード) からのみ、パフォーマンス情報を入手できます。デフォルトでは、デフォルト・ノード (論理ポートが 0 のノード) のパフォーマンス情報が表示されます。他のノードのパフォーマンス情報を表示するには、 DB2NODE 環境変数を、モニターしたいノードのノード番号に設定して、パフォーマンス・モニターの他のセッションを開始する必要があります。

- **DB2 UDB データベース**

  このオブジェクトは、特定のデータベースの情報を提供します。現在アクティブなデータベースごとに、情報を利用できます。

- **DB2 アプリケーション**

  このオブジェクトは、特定の DB2 UDB アプリケーションの情報を提供します。現在アクティブなそれぞれの DB2 UDB アプリケーションに関する情報を利用できます。

- **DB2 DCS データベース**

  このオブジェクトは、特定の DCS データベースの情報を提供します。現在アクティブなデータベースごとに、情報を利用できます。

- **DB2 DCS アプリケーション**

  このオブジェクトは、特定の DB2 DCS アプリケーションの情報を提供します。現在アクティブな DB2 DCS アプリケーションごとに、情報を利用できます。

Windows パフォーマンス・モニターによってリストされるオブジェクトは、Windows マシンに何がインストールされているか、およびどのアプリケーションがアクティブかによって異なります。たとえば、DB2 UDB をインストールしデータベース・マネージャーを開始していれば、 DB2 データベース・マネージャー・オブジェクトがリストされます。さらに、そのマシンで現在アクティブな DB2 UDB データベースおよびアプリケーションがあれば、 DB2 データベースおよび DB2 ア

プリケーション・オブジェクトもリストされます。 Windows システムを DB2 Connect のゲートウェイとして使っていて、現在アクティブな DCS データベースおよびアプリケーションがいくつかある場合、 DB2 DCS データベースおよび DB2 DCS アプリケーション・オブジェクトがリストされます。

関連概念:

- 「*管理ガイド: パフォーマンス*」の『DB2 レジストリー変数と環境変数』

リモートの DB2 UDB パフォーマンス情報へのアクセス

手順:

DB2 パフォーマンス情報にリモートでアクセスできるようにする方法については、すでに説明しました。**「追加先 (Add to)」**ボックスで、モニターする別のコンピューターを選択してください。これにより、そのコンピューター上で使用できるすべてのパフォーマンス・オブジェクトのリストが表示されます。

リモート・コンピューターで DB2 パフォーマンス・オブジェクトをモニターできるようにするには、そのコンピューターにインストールされている DB2 UDB または DB2 Connect コードのレベルが、バージョン 6 以上でなければなりません。

関連概念:

- 413 ページの『Windows パフォーマンス・モニターの紹介』

DB2 パフォーマンス値をリセットする

手順:

アプリケーションで DB2 モニター API を呼び出すと、戻される情報は、通常は DB2 Universal Database™ (DB2 UDB) サーバー開始以降の累積値になります。これは、以下のような場合に役立ちます。

- パフォーマンス値をリセットする
- テストを実行する
- その値をもう一度リセットする
- テストを再実行する

データベースのパフォーマンス値をリセットするには、 **db2perfc** プログラムを使用します。次のように入力してください。

```
db2perfc
```

デフォルトでは、これによりアクティブな DB2 UDB データベースすべてのパフォーマンス値がリセットされます。ただし、リセットするデータベースのリストを指定することも可能です。また -d オプションを使用して、 DCS データベースのパフォーマンス値をリセットするよう指定することもできます。たとえば、以下のとおりです。

```
db2perfc
db2perfc dbalias1 dbalias2 ... dbaliasn
```

```
db2perfc  -d
db2perfc  -d  dbalias1 dbalias2 ... dbaliasn
```

最初の例では、すべてのアクティブな DB2 UDB データベースのパフォーマンス値がリセットされます。次の例では、特定の DB2 UDB データベースの値がリセットされます。 3 番目の例では、すべてのアクティブな DB2 DCS データベースのパフォーマンス値がリセットされます。最後の例では、特定の DB2 DCS データベースの値がリセットされます。

db2perfc プログラムは、関連する DB2 UDB サーバー・インスタンス (つまり、**db2perfc** 実行時のセッションの DB2INSTANCE に保持されるインスタンス) に関するデータベース・パフォーマンス情報に現在アクセスしているすべてのプログラムの値をリセットします。

db2perfc を呼び出すと、 **db2perfc** コマンドの実行中に DB2 UDB パフォーマンス情報にリモートでアクセスしているユーザーがいる場合、そのユーザーに表示される値もリセットされます。

注: sqlmrset という DB2 UDB API を使用すれば、アプリケーションでグローバルではなくローカルに表示される特定のデータベースの値を、アプリケーション側でリセットできます。

関連資料:

- 「*管理 API リファレンス*」の『db2ResetMonitor - モニターのリセット』
- 「*コマンド・リファレンス*」の『db2perfc - データベース・パフォーマンス値のリセット・コマンド』

付録 H. Windows データベース・パーティション・サーバーの使用

Windows 環境で構成の特性を変更する場合、この章で紹介するような特別なユーティリティーを使用します。

この章で紹介する方法は、以下のとおりです。

- 『インスタンス内のデータベース・パーティション・サーバーのリスト』
- 420 ページの『インスタンスへのデータベース・パーティション・サーバーの追加 (Windows)』
- 421 ページの『データベース・パーティションの変更 (Windows)』
- 423 ページの『インスタンスからのデータベース・パーティションのドロップ (Windows)』

インスタンス内のデータベース・パーティション・サーバーのリスト

手順:

Windows で **db2nlist** コマンドを使えば、インスタンスに関与するデータベース・パーティション・サーバーのリストを取得できます。

コマンドは、次のように使用します。

```
db2nlist
```

上記のようにコマンドを使用する場合、デフォルトのインスタンスは (DB2INSTANCE 環境変数によって設定される) 現行インスタンスです。特定のインスタンスを指定するには、次のコマンドを使ってインスタンスを指定できます。

```
db2nlist /i:instName
```

ここで、`instName` は、必要とする特定のインスタンス名です。

次のコマンドを使えば、各パーティション・サーバーの状況を要求することもできます。

```
db2nlist /s
```

各データベース・パーティション・サーバーの状況は、開始中、実行中、停止中、または停止済みのいずれかになります。

関連タスク:

- 420 ページの『インスタンスへのデータベース・パーティション・サーバーの追加 (Windows)』
- 421 ページの『データベース・パーティションの変更 (Windows)』
- 423 ページの『インスタンスからのデータベース・パーティションのドロップ (Windows)』

インスタンスへのデータベース・パーティション・サーバーの追加 (Windows)

手順:

Windows で **db2ncrt** コマンドを使用すると、インスタンスにデータベース・パーティション・サーバー (ノード) を追加できます。

注: このインスタンスの中にデータベースがすでに含まれている場合は、 **db2ncrt** コマンドを使用しないでください。代わりに、**db2start addnode** コマンドを使用します。上記のコマンドにより、新しいデータベース・パーティション・サーバーにデータベースを正しく追加することができます。 `db2nodes.cfg` ファイルは**編集しないでください**。このファイルを変更すると、パーティション・データベース・システムに不整合が生じる可能性があるからです。

このコマンドには、以下の必須パラメーターがあります。

```
db2ncrt /n:node_number
        /u:username,password
        /p:logical_port
```

- /n:

  データベース・パーティション・サーバーを識別するためのユニーク・ノード番号です。番号には、昇順で 1 ～ 999 までを指定できます。

- /u:

  DB2 サービスのログオン・アカウント名とパスワードです。

- /p:logical_port

  論理ポートがゼロ (0) でない場合に、データベース・パーティション・サーバーに使用する論理ポート番号。このパラメーターを指定しないと、論理ポート番号には 0 が割り当てられます。

マシンに最初のノードを作成するときには、論理ポート・パラメーターはオプショナルです。論理ノードを作成する場合は、このパラメーターを指定し、未使用の論理ポート番号を選択しなければなりません。このパラメーターの使用に関して、いくつかの制約事項があります。

- どのマシンにも、論理ポート 0 のデータベース・パーティション・サーバーが 1 つずつ存在しなければなりません。
- `%SystemRoot%¥system32¥drivers¥etc` ディレクトリーのサービス・ファイル内で、 FCM 通信のために予約されているポート範囲よりも大きいポート番号を選択することはできません。たとえば、現行インスタンスのために 4 つのポートの範囲を予約する場合、最大ポート番号は 3 になるはずです (ポート 1、2、3。ポート 0 はデフォルトの論理ノード)。ポート範囲は、 **db2icrt** を `/r:base_port,end_port` パラメーターと一緒に使用するときに定義します。

次のような任意指定パラメーターもあります。

- /g:network_name

データベース・パーティション・サーバーのネットワーク名を指定します。このパラメーターを指定しなかった場合、 DB2 はシステムで最初に検出した IP アドレスを使用します。

マシン上に複数の IP アドレスがあり、データベース・パーティション・サーバーに特定の IP アドレスを割り当てたい場合に、このパラメーターを使用します。 ネットワーク名や IP アドレスを *network_name* パラメーターに入力することができます。

- /h:host_name

  TCP/IP ホスト名。ホスト名がローカル・ホスト名でない場合に FCM が内部通信用に使用します。このパラメーターが必要になるのは、データベース・パーティション・サーバーをリモート・マシンに追加する場合です。

- /i:instance_name

  インスタンス名。デフォルトは、現行インスタンスです。

- /m:machine_name

  ノードが常駐する Windows ワークステーションのコンピューター名。デフォルトの名前は、ローカル・マシンのコンピューター名です。

- /o:instance_owning_machine

  インスタンス所有マシンであるマシンのコンピューター名。デフォルトはローカル・マシンです。このパラメーターは、インスタンス所有マシンでないマシンで **db2ncrt** コマンドを呼び出すときに必須です。

たとえば、(複数の論理ノードを実行するために) 新しいデータベース・パーティション・サーバーを、インスタンス所有マシン MYMACHIN 上のインスタンス TESTMPP へ追加して、この新しいノードを論理ポート 1 を使用するノード 2 として認識されるようにするには、次のように入力します。

```
db2ncrt /n:2 /p:1 /u:my_id,my_pword /i:TESTMPP
   /M:TEST /o:MYMACHIN
```

関連資料:

- 「コマンド・リファレンス」の『db2start - DB2 の開始コマンド』
- 「コマンド・リファレンス」の『db2icrt - インスタンスの作成コマンド』
- 「コマンド・リファレンス」の『db2ncrt - インスタンスへのデータベース・パーティション・サーバーの追加コマンド』

データベース・パーティションの変更 (Windows)

手順:

Windows で **db2nchg** コマンドを使用して、以下を行うことができます。

- データベース・パーティションをあるマシンから別のマシンに移動する。
- マシンの TCP/IP ホスト名を変更する。

複数のネットワーク・アダプターを使用する予定の場合には、このコマンドを使用して、 *db2nodes.cfg* ファイルの ″netname″ フィールドに TCP/IP アドレスを指定しなければなりません。

- 異なる論理ポート番号を使用する。
- データベース・パーティション・サーバー (ノード) に異なる名前を使用する。

このコマンドには、以下の必要パラメーターがあります。

```
db2nchg /n:node_number
```

パラメーター /n: は、構成を変更したいデータベース・パーティション・サーバーのノード番号です。このパラメーターは必須です。

オプション・パラメーターには、以下のものがあります。

- /i:instance_name

  このデータベース・パーティション・サーバーが参加しているインスタンスを指定します。このパラメーターを指定しなかった場合、デフォルトである現行インスタンスが使用されます。

- /u:username,password

  DB2 Universal Database™ (DB2 UDB) サービスのログオン・アカウント名とパスワードを変更します。このパラメーターを指定しなかった場合、ログオン・アカウント名とパスワードは変わりません。

- /p:logical_port

  データベース・パーティション・サーバーの論理ポートを変更します。データベース・パーティション・サーバーを異なるマシンへ移動させる場合、このパラメーターの指定は必須です。このパラメーターを指定しなかった場合、論理ポート番号は変わりません。

- /h:host_name

  FCM が内部通信のために使用する TCP/IP ホスト名を変更します。このパラメーターを指定しなかった場合、ホスト名は変わりません。

- /m:machine_name

  データベース・パーティション・サーバーを別のマシンへ移動させます。データベース・パーティション・サーバーを移動できるのは、インスタンス内にデータベースが 1 つもない場合だけです。

- /g:network_name

  データベース・パーティション・サーバーのネットワーク名を変更します。

  マシン上に複数の IP アドレスがあり、データベース・パーティション・サーバーに特定の IP アドレスを割り当てたい場合に、このパラメーターを使用します。ネットワーク名や IP アドレスを network_name に入力することができます。

たとえば、ノード 2 に割り当てられている論理ポート (インスタンス TESTMPP に参加している) が論理ポート 3 を使用するように変更したい場合は、次のコマンドを入力します。

```
db2nchg /n:2 /i:TESTMPP /p:3
```

DB2 UDB には、リモート・マシン上のインスタンス・レベルの DB2 UDB レジストリー変数にアクセスする機能があります。現在、DB2 UDB レジストリー変数は、マシンまたはグローバル・レベル、インスタンス・レベル、およびノード・レベルの 3 つの異なるレベルに保管されています。インスタンス・レベル (ノード・レベルも含む) に保管されているレジストリー変数は、 DB2REMOTEPREG を使用して別のマシンにリダイレクトすることができます。 DB2REMOTEPREG が設定されると、DB2 UDB は、 DB2REMOTEPREG が示すマシンから DB2 UDB レジストリー変数にアクセスします。 db2set コマンドは、次のようになります。

```
db2set DB2REMOTEPREG=<remote workstation>
```

ここで <remote workstation> は、リモート・ワークステーション名です。

注: すべての DB2 UDB インスタンス・プロファイルとインスタンス・リストは、指定されたリモート・マシン名に格納されるため、このオプションの設定には注意が必要です。

この機能は、レジストリーが含まれる同じマシン上のリモート LAN 装置を指すために、 DBINSTPROF の設定と組み合わせて使用することができます。

関連概念:

- 「*管理ガイド: パフォーマンス*」の『DB2 レジストリー変数と環境変数』

関連資料:

- 「コマンド・リファレンス」の『db2nchg - データベース・パーティション・サーバー構成の変更コマンド』

インスタンスからのデータベース・パーティションのドロップ (Windows)

手順:

Windows で **db2ndrop** コマンドを使うと、データベースのないインスタンスからデータベース・パーティション・サーバー (ノード) をドロップできます。データベース・パーティション・サーバーをドロップする場合、そのノード番号は新しいデータベース・パーティション・サーバーに再使用することができます。

インスタンスからデータベース・パーティション・サーバーをドロップする場合は、注意してください。インスタンス所有データベース・パーティション・サーバーのノードのゼロ (0) をインスタンスからドロップすると、インスタンスは使用できなくなります。インスタンスをドロップしたい場合は、**db2idrop** コマンドを使用します。

注: このインスタンスの中にデータベースが含まれている場合は、 **db2ndrop** コマンドを使用しないでください。代わりに、**db2stop drop nodenum** コマンドを使用します。上記のコマンドにより、新しいデータベース・パーティションからデータベースを正しく除去することができます。 db2nodes.cfg ファイル

は**編集しないでください**。このファイルを変更すると、パーティション・データベース・システムに不整合が生じる可能性があるからです。

複数の論理ノードが実行されているマシンから、論理ポート 0 に割り当てられているノードをドロップする場合は、 0 以外の論理ポートに割り当てられているノードをすべてドロップしてからでないと、論理ポート 0 に割り当てられているノードをドロップできません。どのデータベース・パーティション・サーバーにも、論理ポート 0 に割り当てられているノードが 1 つずつなければなりません。

このコマンドには、以下のパラメーターがあります。

```
db2ndrop /n:node_number /i:instance_name
```

- /n:

  データベース・パーティション・サーバーを識別するためのユニーク・ノード番号です。これは必要パラメーターです。番号には、昇順でゼロ (0) から 999 までを指定できます。ノードのゼロ (0) はインスタンス所有マシンを表すことに注意してください。

- /i:instance_name

  インスタンス名です。これはオプション・パラメーターです。このパラメーターを指定しない場合、デフォルトのインスタンスは (DB2INSTANCE レジストリー変数によって設定される) 現行インスタンスです。

関連概念:

- 「*管理ガイド: パフォーマンス*」の『DB2 レジストリー変数と環境変数』

関連資料:

- 「コマンド・リファレンス」の『db2stop - DB2 の停止コマンド』
- 「コマンド・リファレンス」の『db2idrop - インスタンスの除去コマンド』
- 「コマンド・リファレンス」の『db2ndrop - インスタンスからのデータベース・パーティション・サーバーのドロップ・コマンド』

付録 I. 複数の論理データベース・パーティションの構成

複数の論理データベース・パーティションを使用する状況

DB2® Universal Database (DB2 UDB) Enterprise Server Edition では、各マシンにデータベース・パーティション・サーバーが 1 つずつ割り当てられるように構成するのが一般的です。ただし、状況によっては、複数のデータベース・パーティション・サーバーを同一のマシンで実行したほうがよい場合もあります。つまり、構成にマシンの数よりも多くのデータベース・パーティションが含まれることがあります。その場合、それらのデータベース・パーティションが*同じ* インスタンスに参加しているのであれば、そのマシンでは*複数*のデータベース・パーティション が実行されているといいます。それらのデータベース・パーティションが異なる複数のインスタンスに参加している場合には、このマシンは複数の論理データベース・パーティションをホスティングしていません。

複数の論理データベース・パーティションがサポートされているため、次のような 3 種類の構成の中から選ぶことができます。

- 各マシンにデータベース・パーティション・サーバーが 1 つずつ用意されている、標準的な構成。
- 1 台のマシンに複数のデータベース・パーティション・サーバーがある、複数論理データベース・パーティション構成。
- 複数のマシンのそれぞれで複数の論理データベース・パーティションが実行される構成。

複数の論理ノードを使用する構成は、対称マルチプロセッサー (SMP) アーキテクチャーのマシン上でシステムが照会を実行するときに便利です。さらに、1 つのマシンに複数の論理データベース・パーティションを構成すると、いずれかのマシンで障害が発生した場合にも効果を発揮します。あるマシンで障害が発生しても (その結果、そのマシン上の 1 つまたは複数のデータベース・パーティション・サーバーが使用できなくなっても)、 DB2START NODENUM コマンドを使用すれば、他のマシンでそのデータベース・パーティション・サーバーを再始動できます。これにより、ユーザー・データを確実に引き続き利用できます。

もう 1 つの利点は、複数の論理データベース・パーティションがあれば SMP ハードウェア構成を十分に活用できることです。さらに、データベース・パーティション・サーバーが小さくなるので、データベース・パーティションや表スペースのバックアップとリストア、索引の作成などのタスクを実行するときのパフォーマンスが向上します。

関連タスク:

- 426 ページの『複数の論理データベース・パーティションの構成』

関連資料:

- 「*コマンド・リファレンス*」の『db2start - DB2 の開始コマンド』

複数の論理データベース・パーティションの構成

手順:

次の 2 つの方法のいずれかで複数の論理データベース・パーティションを構成できます。

- db2nodes.cfg ファイル内で論理データベース・パーティション (ノード) を構成します。構成後、DB2START コマンドとその関連 API を使用して、すべての論理データベース・パーティションとリモート・データベース・パーティションを開始できます。

  注: Windows NT 環境では、システム内にデータベースが 1 つもない場合は、*db2ncrt* を使用してデータベース・パーティションを追加する必要があります。 1 つまたは複数のデータベースがある場合は、 DB2START ADDNODE コマンドを使用します。 Windows NT 内では、 *db2nodes.cfg* ファイルを手動で編集することは絶対に避けてください。

- 他の論理データベース・パーティション (ノード) がすでに実行している、別のプロセッサー上で論理データベース・パーティションを再始動します。この場合、db2nodes.cfg 内で論理データベース・パーティションとして指定したホスト名とポート番号をオーバーライドできます。

db2nodes.cfg 内で論理データベース・パーティション (ノード) を構成するには、このファイル内に項目を作成して、データベース・パーティションの論理ポート番号を割り当てなければなりません。次の構文を使用する必要があります。

```
nodenumber hostname logical-port netname
```

注: Windows NT 環境では、システム内にデータベースが 1 つもない場合は、*db2ncrt* を使用してデータベース・パーティションを追加する必要があります。 1 つまたは複数のデータベースがある場合は、 DB2START ADDNODE コマンドを使用します。 Windows NT 内では、 *db2nodes.cfg* ファイルを手動で編集することは絶対に避けてください。

Windows NT での *db2nodes.cfg* ファイルの形式は、 UNIX での同じファイルとは異なります。 Windows NT での列形式は、次のとおりです。

```
nodenumber hostname computername logical_port netname
```

ホスト名には、完全修飾名を使用します。 /etc/hosts ファイルも、完全修飾名を使用する必要があります。 db2nodes.cfg ファイルおよび /etc/hosts ファイルで完全修飾名を使用しない場合、エラー・メッセージ SQL30082N RC=3 を受け取る場合があります。

FCM 通信にとって十分な数のポートを etc ディレクトリーの services ファイルに定義しなければなりません。

関連概念:

- 425 ページの『複数の論理データベース・パーティションを使用する状況』

関連タスク:

- 166 ページの『ノードおよびデータベース構成ファイルの変更』

- 39 ページの『ノード構成ファイルの作成』

関連資料:

- 「コマンド・リファレンス」の『db2start - DB2 の開始コマンド』
- 「コマンド・リファレンス」の『db2ncrt - インスタンスへのデータベース・パーティション・サーバーの追加コマンド』

付録 J. コントロール・センターの拡張

コントロール・センター用のプラグイン・アーキテクチャーの紹介

新しいプラグイン・アーキテクチャーを使って機能を追加することにより、 DB2 Universal Database のコントロール・センターを拡張することができます。

このプラグイン・アーキテクチャーの目的は、「コントロール・センター」ポップアップ・メニューにある所定のオブジェクトに項目を追加したり、「コントロール・センター」ツリーにオブジェクトを追加したり、ツールバーに新しいボタンを追加する機能を提供することです。インプリメントしなければならない一連の Java™ インターフェースが、それらのツールに付属しています。これらのインターフェースは、どのアクションを追加するかに関してコントロール・センターと通信するために使用されます。

プラグイン拡張機能 (db2plug.zip) は、コントロール・センター・ツールの起動時にロードされます。そのため、ZIP ファイルのサイズによっては、ツールの起動時間が長くなる可能性があります。しかし、プラグインの ZIP ファイルはほとんどのユーザーにとって小さく、その影響は最小限に抑えられるはずです。

関連概念:

- 430 ページの『サンプル・プラグインのコンパイルおよび実行』
- 432 ページの『コントロール・センターの拡張機能としてのプラグインの作成』
- 429 ページの『コントロール・センター用プラグインの開発者向けのガイドライン』

コントロール・センター用プラグインの開発者向けのガイドライン

db2plug.zip ファイルには複数のプラグインを入れることができるので、プラグインの開発者は、コントロール・センター用のプラグインを作成する場合は、以下のガイドラインに従う必要があります。

- Java™ パッケージを使用して、プラグイン・クラスがユニークな名前を持つようにする。 Java パッケージの命名規則に従ってください。パッケージ名の接頭部には、インターネット・ドメインを逆にした名前 (たとえば、 com.companyname) を付けます。パッケージ名の全部または少なくともユニークな接頭部は、小文字でなければなりません。
- db2plug.zip は、 `sqllib` ディレクトリーの下の tools ディレクトリーにインストールする必要があります。 V8 以前では、db2plug.zip は、 `sqllib` ディレクトリーの下の `cc` ディレクトリーにインストールする必要がありました。
- コントロール・センター用のプラグインを作成する場合で、 db2plug.zip ファイルがすでに存在する場合は、プラグイン・クラスを既存の db2plug.zip に追加する必要があります。既存の db2plug.zip ファイルを、独自の db2plug.zip ファイルで上書きしてはなりません。既存の db2plug.zip にプラグインを追加するには、以下の zip コマンドを使用してください。

```
zip -r0 db2plug.zip com¥companyname¥myplugin¥*.class
```

プラグイン・パッケージ名は com.companyname.myplugin です。

- db2plug.zip 内のすべてのクラスは、コントロール・センターの開始時にロードされます。 db2plug.zip ファイルには、com.ibm.db2.tools.cc.navigator パッケージ内のクラスを拡張またはインプリメントする、すべての CCExtension クラス・ファイルおよびクラスが含まれていなければなりません。これらのクラスによって直接使用されないクラスは、 db2plug.zip に含まれている必要はありません。これらのファイルは、コントロール・センターの開始時のパフォーマンス上の影響を最小限に抑えるために、別の jar ファイルに保管することができます。これは、余分のクラスが多数ある場合に役立ちます。 jar ファイルは、sqllib ディレクトリーの下の tools ディレクトリーに置く必要があります。 **db2cc** コマンドを使用してコントロール・センターを開始すると、 jar ファイルは自動的に *classpath* に組み込まれます。
- CCObject をインプリメントするプラグイン・クラスは、コントロール・センターから Class.newInstance() への呼び出しが可能な、引き数なしのデフォルト・コンストラクターを提供する必要があります。
- 内部クラスの使用を可能な限り避けます。一般的に、コントロール・センターで新規プラグイン・オブジェクトを作成するために、 CCTreeObject をインプリメントするプラグイン・クラスを、内部クラスとして宣言することはできません。これは、コントロール・センターがこれらのクラスをインスタンス化するのを防ぎます。
- **db2cc -tf filename** を使用して、プラグインが正常にロードされているかをテストします。これにより、コントロール・センターのトレース情報が指定したファイル名で書き込まれます。絶対パス名を指定しないと、トレース・ファイルは sqllib のツール・ディレクトリーに書き込まれます。プラグイン関連のトレース・ステートメントには、「Plugin」という語が含まれます。「PluginLoader」というテキストが含まれる行を探すことで、クラスがロードされているかどうかを確認できます。

関連概念:

- 430 ページの『サンプル・プラグインのコンパイルおよび実行』
- 432 ページの『コントロール・センターの拡張機能としてのプラグインの作成』

関連資料:

- 「コマンド・リファレンス」の『db2cc - コントロール・センターの開始コマンド』

サンプル・プラグインのコンパイルおよび実行

コントロール・センター用のプラグインの機能は、これ以降のセクションと、対応するプラグイン・サンプル・プログラム (Example1.java、 Example2.java、 Example3.java、 Example3Folder.java、および Example3Child.java) に例示されています。これらのサンプルの Java ファイルは、 DB2® Application Development Client と一緒にインストールされます。 Windows® プラットフォームでは、これらのサンプル・プログラムは、 DRIVE:¥sqllib¥samples¥java¥plugin にあります。ここで DRIVE: は、DB2 のインストール先のドライブを表します。 UNIX® プラットフォ

ームでは、これらのサンプルは、 /u/db2inst1/sqllib/samples/java/plugin にあります。ここで /u/db2inst1 は、 DB2 のインストール先のディレクトリーを表します。

注: プラグイン・サンプル・プログラムには、以下にまだ反映されていない更新が含まれている可能性があります。サンプル・コードおよび Java ドキュメンテーションが以下の情報と異なる場合、前者が最新の情報です。

サンプルのプラグインを実行するには、Java™ アーカイブ・ファイルの規則に従って、拡張クラス・ファイルを ZIP する必要があります。 ZIP ファイル (db2plug.zip) は *classpath* 内に置いておかなければなりません。 Windows オペレーティング・システムでは、db2plug.zip を DRIVE:¥sqllib¥tools ディレクトリーに置いてください。ここで DRIVE: は、DB2 のインストール先のドライブを表します。UNIX プラットフォームでは、db2plug.zip を /u/db2inst1/sqllib/tools ディレクトリーに置いてください。ここで /u/db2inst1 は、DB2 のインストール先のディレクトリーを表します。

注: **db2cc** コマンドは、 *classpath* を tools ディレクトリー内の db2plug.zip を指すように設定します。

サンプル (Example3、Example3Folder および Example3Child を除く) は、互いに競合する場合があるので同じ db2plug.zip に ZIP してはなりません。

これらのサンプルの Java ファイルをコンパイルするには、以下を *classpath* に組み込む必要があります。

- Windows プラットフォームでは、以下を使用します。
  - DRIVE: ¥sqllib¥java¥Common.jar
  - DRIVE:¥sqllib¥tools¥db2navplug.jar

  ここで DRIVE は、DB2 のインストール先のドライブを表します。
- UNIX プラットフォームでは、以下を使用します。
  - /u/db2inst1/sqllib/java/Common.jar
  - /u/db2inst1/sqllib/tools/db2navplug.jar

  ここで /u/db2inst1 は、DB2 のインストール先のディレクトリーを表します。

サンプル Java ファイルをコンパイルして生成されるすべてのクラスを組み込むために、 db2plug.zip を作成します。このファイルを圧縮しないでください。たとえば、以下を発行します。

```
zip -r0 db2plug.zip *.class
```

このコマンドを発行すると、すべてのクラス・ファイルが db2plug.zip ファイルに入れられ、相対パス情報が保存されます。

関連概念:

- 432 ページの『コントロール・センターの拡張機能としてのプラグインの作成』
- 429 ページの『コントロール・センター用プラグインの開発者向けのガイドライン』

関連資料:

- 「コマンド・リファレンス」の『db2cc - コントロール・センターの開始コマンド』

コントロール・センターの拡張機能としてのプラグインの作成

プラグインを作成するための最初のステップは、CCExtension インターフェースをインプリメントするクラスを定義することです。このクラスには、コントロール・センターによってロードされるプラグイン・クラスのリストが含まれます。メニュー項目を Database や Table などの標準のコントロール・センター・オブジェクトに追加するか、ツリーに表示する独自のオブジェクトを作成する場合、CCObject インターフェースをインプリメントし、 getObjects メソッドでこれらの CCObject の配列を戻すクラスを作成します。ツールバー・ボタンを追加する場合、CCToolbarAction をインプリメントし、 getToolbarActions メソッドで CCToolbarActions の配列を戻します。

それぞれのインターフェースについては、以下のファイルで説明されています。

- Windows® プラットフォームの場合、`DRIVE:¥sqllib¥samples¥java¥plugin¥doc`。`DRIVE:` は DB2® がインストールされているドライブです。
- UNIX® プラットフォームの場合、`/u/db2inst1/sqllib/samples/java/plugin/doc` です。 `/u/db2inst1` は DB2 がインストールされているディレクトリーです。

関連タスク:

- 433 ページの『ツールバー・ボタンを追加するプラグインの作成』
- 434 ページの『基本メニュー・アクションの作成』
- 436 ページの『メニュー項目の配置』
- 437 ページの『基本メニュー・アクションの区切り記号の作成』
- 437 ページの『サブメニューの作成』
- 438 ページの『特定の名前を持つオブジェクトにだけメニュー項目を追加する』
- 439 ページの『ツリーに複数のオブジェクトを入れるためのフォルダーの追加』
- 441 ページの『フォルダーの下へのサンプル・オブジェクトの追加』
- 442 ページの『プラグイン・ツリー・オブジェクトの属性の設定』
- 444 ページの『Create アクションの追加』
- 445 ページの『複数選択をサポートする Remove アクションの追加』
- 447 ページの『Alter アクションの追加』
- 448 ページの『isConfigurable() を使用して構成機能を使用不可にする』
- 448 ページの『isEditable() を使用してオブジェクト変更機能を使用不可にする』
- 449 ページの『hasConfigurationDefaults() を使用して構成ダイアログのデフォルト・ボタンを使用不可にする』

プラグイン・タスクの説明

以下のプラグイン・タスクについて説明します。

1. ツールバー・ボタンを追加するプラグインの作成
2. Database オブジェクトに新規のメニュー項目を追加するプラグインの作成

3. ツリーで Database の下にプラグイン・オブジェクトを追加するプラグインの作成
4. isConfigurable() を使用して構成機能を使用不可にする
5. isEditable() を使用してオブジェクト変更機能を使用不可にする
6. hasConfigurationDefaults() を使用して構成ダイアログのデフォルト・ボタンを使用不可にする

ツールバー・ボタンを追加するプラグインの作成

手順:

この例では、ツールバー・ボタンを追加するだけなので、次のようにして、getObjects は NULL 配列を戻す必要があります。

```
import  com.ibm.db2.tools.cc.navigator.*;
import  java.awt.event.*;
import  javax.swing.*;

public class Example1 implements CCExtension {

  public CCObject[] getObjects () {
    return null;
  }

}
```

com.ibm.db2.tools.cc.navigator パッケージがインポートされることに注意してください。このクラスは、getHoverHelpText、getIcon、および actionPerformed の 3 つのメソッドをインプリメントする必要がある、 CCToolbarAction インターフェースをインプリメントします。コントロール・センターは getHoverHelpText を使用することにより、テキストを含む小さなボックスを表示します。これは、ユーザーがツールバー・ボタンの上にマウスを置いて吹き出しを表示させるときに示されるものです。 getIcon を使用して、ボタンのアイコンを指定します。コントロール・センターは、ユーザーがボタンをクリックするときに、actionPerformed を呼び出します。これは、メッセージを書き込む X というボタンをクリックすると、そのボタンをコンソールに追加する例です。ここでは、コントロール・センターのイメージ・リポジトリー・クラスの「リフレッシュ (Refresh)」アイコンを使用します。

```
class Example1ToolbarAction implements CCToolbarAction {

  public String getHoverHelpText() { return "X"; }

  public ImageIcon getIcon() {
   return CommonImageRepository.getCommonIcon(CommonImageRepository.WC_NV_
REFRESH);
  }

  public void actionPerformed(ActionEvent e) {
   System.out.println("I've been clicked");
  }

}
```

最後のステップでは、Example1 に getToolbarActions メソッドをインプリメントし、次のようにして、新しいクラスのインスタンスを戻します。

```
  public CCToolbarAction[] getToolbarActions () {
   return new CCToolbarAction[] { new Example1ToolbarAction() };
  }
```

関連概念:

- 430 ページの『サンプル・プラグインのコンパイルおよび実行』

Database オブジェクトに新規のメニュー項目を追加するプラグインの作成

以下の手順は、 Database オブジェクトに新規のメニュー項目を追加するプラグインを作成する方法を示しています。

1. 基本メニュー・アクションの作成
2. メニュー項目の配置
3. 基本メニュー・アクションの区切り記号の作成
4. サブメニューの作成
5. 特定の名前を持つオブジェクトにだけメニュー項目を追加する

基本メニュー・アクションの作成

手順:

このより詳しいトピックでは、新しいコマンドを Database オブジェクトのポップアップ・メニューに追加します。

例 1 にあるように、最初のステップは、CCExtension を拡張するクラスの作成です。

```
import  com.ibm.db2.tools.cc.navigator.*;
import  java.awt.event.*;
import  javax.swing.*;

public class Example2 implements CCExtension {

  public CCToolbarAction[] getToolbarActions () {
    return null;
  }

}
```

2 番目のステップは、次のようにして、ツリー内の Database オブジェクト用に CCObject を作成することです。

```
  class CCDatabase implements CCObject {

    public String getName () { return null; }
    public boolean isEditable () { return true; }
    public boolean isConfigurable () { return true; }

    public int getType () { return UDB_DATABASE; }
  }
```

コントロール・センターの組み込みオブジェクト (たとえばこの例では、Database オブジェクト) にメニュー項目を追加する機能以外に、機能を使用する予定はないため、NULL または TRUE を戻すようにほとんどの関数をインプリメントします。このオブジェクトが DB2 UDB Database オブジェクトを表すように指定するには、

そのタイプを UDB_DATABASE (つまり CCObject の定数) として指定します。クラスの名前は、この例では CCDatabase ですが、使用するプラグインと同じ zip ファイル名のベンダー製プラグインが存在する可能性がありますので、クラス名はできる限りユニークなものにしてください。確実にユニークなクラス名を使用するようにするには、Java パッケージを使用する必要があります。

CCExtension の getObjects メソッドは、次のようにして、 CCDatabase のインスタンスを含む配列を戻すようになります。

```
public CCObject[] getObjects () {
  return new CCObject[] { new CCDatabase() };
}
```

タイプが UDB_DATABASE である複数の CCObject サブクラスを作成できますが、 isEditable または isConfigurable メソッドから戻される値が矛盾する場合には、 false を戻すオブジェクトは true を戻すオブジェクトをオーバーライドします。

インプリメントする残りのメソッドは、getMenuActions だけです。これは、CCMenuActions の配列を戻すので、まずこのインターフェースをインプリメントするクラスを作成します。

インプリメントするメソッドは、getMenuText と actionPerformed の 2 つです。メニューに表示されるテキストは、getMenuText を使用して入手します。ユーザーがメニュー項目をクリックすると、生成されるイベントにより、 actionPerformed への呼び出しが行われます。

次のクラス例では、1 つのデータベース・オブジェクトが選択されると、″Example2a Action″ というメニュー項目が表示されます。ユーザーがこのメニュー項目をクリックすると、コンソールにメッセージ ″Example2a menu item actionPerformed″ が示されます。

```
class Example2AAction implements CCMenuAction {

  public String getMenuText () { return "Example2a Action"; }

  public void actionPerformed (ActionEvent e) {
    System.out.println("Example2a menu item actionPerformed");
  }

}
```

最後に、CCObject に以下のとおり追加することにより、このメニュー項目を UDB データベースの CCObject に追加します。

```
public CCMenuAction[] getMenuActions () {
     return new CCMenuAction[] { new Example2AAction() };
  }
```

関連概念:

- 430 ページの『サンプル・プラグインのコンパイルおよび実行』

関連タスク:

- 436 ページの『メニュー項目の配置』
- 437 ページの『基本メニュー・アクションの区切り記号の作成』
- 437 ページの『サブメニューの作成』

- 438 ページの『特定の名前を持つオブジェクトにだけメニュー項目を追加する』

メニュー項目の配置

手順:

基本メニュー項目を作成するときには、メニュー内でのメニュー項目の位置を指定しませんでした。プラグイン・メニュー項目をメニューに追加するときのデフォルトの動作として、メニュー項目を最後に追加しますが、「リフレッシュ (Refresh)」および「フィルター (Filter)」メニュー項目があれば、その前に追加します。

この動作を変更し、ゼロからメニュー内の項目数までの位置番号 (ただし、「リフレッシュ (Refresh)」および「フィルター (Filter)」メニュー項目はカウントしない) を指定することができます。 Positionable をインプリメントするように CCMenuAction サブクラスを変更してから、次のようにして getPosition メソッドをインプリメントします。

```
class Example2BAction implements CCMenuAction, Positionable {

  public String getMenuText () { return "Example2B Action"; }

  public void actionPerformed (ActionEvent e) {
    System.out.println("Example2B menu item actionPerformed");
  }

  public int getPosition() {
     return 0;
  }
}
```

ゼロの位置番号を指定すると、メニュー項目はリストの最初に配置され、プラグイン・メニュー項目をカウントしないメニューの項目数と同じ位置番号を指定すると、リストの最後 (ただし、「リフレッシュ (Refresh)」および「フィルター (Filter)」メニュー項目の前) に配置されます。また、Positionable.POSITION_BOTTOM の値を戻してデフォルトの動作を入手することも可能です。つまり、メニュー項目を最後 (ただし、「リフレッシュ (Refresh)」および「フィルター (Filter)」メニュー項目の前) に配置できます。メニュー項目の位置が POSITION_BOTTOM である、タイプ UDB_DATABASE の CCObject が複数存在する場合、メニュー項目は、タイプ UDB_DATABASE の CCObject が CCExtension の getObjects メソッドで戻される順序に基づいて並べられます。

次のようにして、Example2BAction をメニューに追加するよう CCDatabase を変更します。

```
public CCMenuAction[] getMenuActions () {
  return new CCMenuAction[] { new Example2AAction(),
                              new Example2BAction() };
}
```

関連タスク:

- 434 ページの『基本メニュー・アクションの作成』
- 437 ページの『基本メニュー・アクションの区切り記号の作成』
- 437 ページの『サブメニューの作成』
- 438 ページの『特定の名前を持つオブジェクトにだけメニュー項目を追加する』

基本メニュー・アクションの区切り記号の作成

手順:

区切り記号を追加するには、Separator インターフェースをインプリメントする CCMenuAction を作成します。他のメソッド (Positionable をインプリメントする場合は getPosition を除く) は、すべて無視されます。

```
class Example2CSeparator implements CCMenuAction, Separator, Positionable {

  public String getMenuText () { return null; }

  public void actionPerformed (ActionEvent e) {}

  public int getPosition() {
     return 1;
  }
}

public CCMenuAction[] getMenuActions () {
  return new CCMenuAction[] { new Example2AAction(),
                              new Example2BAction(),
                              new Example2CSeparator() };
}
```

関連タスク:

- 434 ページの『基本メニュー・アクションの作成』
- 436 ページの『メニュー項目の配置』
- 437 ページの『サブメニューの作成』
- 438 ページの『特定の名前を持つオブジェクトにだけメニュー項目を追加する』

サブメニューの作成

手順:

サブメニューとは、CCMenuActions の配列のことです。メニュー項目にサブメニューを含めるには、SubMenuParent インターフェースをインプリメントする必要があります。その後、サブメニュー項目ごとに CCMenuAction のインプリメンテーションを作成し、それを SubMenuParent インターフェースの getSubMenuActions メソッドからの配列に戻します。メニュー項目をプラグイン以外のサブメニューに追加することはサポートされていません。また、SubMenuParent は、コントロール・センターから ActionEvents を受け取らない点にも注意してください。以下はその例です。

```
class Example2DAction implements CCMenuAction, SubMenuParent {

  public String getMenuText () { return "Example2D Action"; }

  public void actionPerformed (ActionEvent e) {}

  public CCMenuAction[] getSubMenuActions() {
     return new CCMenuAction[] { new Example2DSubMenuAction() };
  }

}

class Example2DSubMenuAction implements CCMenuAction {

  public String getMenuText () { return "Example2D Sub-Menu Action"; }
```

```
    public void actionPerformed (ActionEvent e) {
      System.out.println("Example2D sub-menu menu item actionPerformed");
    }

  }
```

もう一度、この新しいメニュー項目を CCDatabase に追加します。

```
    public CCMenuAction[] getMenuActions () }
      return new CCMenuAction[] { new Example2AAction(),
                                  new Example2BAction(),
                                  new Example2CSeparator(),
                                  new Example2DAction() };
    }
```

関連タスク:

- 434 ページの『基本メニュー・アクションの作成』
- 436 ページの『メニュー項目の配置』
- 437 ページの『基本メニュー・アクションの区切り記号の作成』
- 438 ページの『特定の名前を持つオブジェクトにだけメニュー項目を追加する』

特定の名前を持つオブジェクトにだけメニュー項目を追加する

手順:

現在のところ、作成済みのプラグイン・メニュー項目は、コントロール・センターに表示されるすべてのデータベースに示されます。 CCDatabase の getName で特定データベースの名前を戻すことにより、このようなメニュー項目を特定の名前のデータベースだけに限定できます。これは、完全修飾名でなければなりません。ここではデータベースを参照しているため、getName メソッドで戻す内容には、システム名、インスタンス名、およびデータベース名を含める必要があります。これらの名前は、″ - ″ で区切ります。以下は、MYSYSTEM というシステム、DB2 というインスタンス、および SAMPLE というデータベースの場合の例です。

```
class CCDatabase implements CCObject {
  ...
  public String getName () { return "MYSYSTEM - DB2 - SAMPLE"; }
  ...
}
```

関連タスク:

- 434 ページの『基本メニュー・アクションの作成』
- 436 ページの『メニュー項目の配置』
- 437 ページの『基本メニュー・アクションの区切り記号の作成』
- 437 ページの『サブメニューの作成』

ツリーで Database の下にプラグイン・オブジェクトを追加するプラグインの作成

以下の手順は、ツリーで Database の下にプラグイン・オブジェクトを追加するプラグインを作成する方法を示しています。

1. ツリーに複数のオブジェクトを入れるためのフォルダーの追加
2. フォルダーの下へのサンプル・オブジェクトの追加

3. プラグイン・ツリー・オブジェクトの属性の設定
4. Create アクションの追加
5. Remove アクションの追加
6. Alter アクションの追加

ツリーに複数のオブジェクトを入れるためのフォルダーの追加

手順:

この例では、プラグイン・オブジェクトを「コントロール・センター (Control Center)」ツリーの「データベース (Database)」の下に示すため、 CCObject ではなく CCTreeObject をインプリメントします。まず、このオブジェクトのために CCTreeObject インプリメンテーションを作成する必要があります。ツリーに配置するオブジェクトが複数存在する場合は、「データベース (Database)」の下にすべてを直接配置するのではなく、通常はフォルダーを作成します。以下は、初期バージョンのフォルダーです。

```
  public class Example3Folder implements CCTreeObject {
    private String parentName = null;
    public boolean isEditable () { return false; }
    public boolean isConfigurable () { return false; }
    public CCTableObject getChildren () { return null; }
    public void setParentName(String name)
    {
      parentName = name;
    }
    public CCColumn[] getColumns () { return null; }
    public boolean isLeaf () { return false; }
    public CCMenuAction[] getMenuActions () { return null; }

    public String getName () { return "Example3 Folder"; }

    public void getData (Object[] data) {
       data[0] = this;
    }

    public int getType () { return CCTypeFactory.getTypeNumber
(this.getClass().getName()); }

    public Icon getIcon (int iconState) {
      switch (iconState) {
        case CLOSED_FOLDER:
  return CommonImageRepository.getScaledIcon(CommonImageRepository.NV_CLOSED_
FOLDER);

        case OPEN_FOLDER:
  return CommonImageRepository.getScaledIcon(CommonImageRepository.NV_OPEN_
FOLDER);

        default:
  return CommonImageRepository.getScaledIcon(CommonImageRepository.NV_CLOSED_
FOLDER);
      }
    }

  }
```

getType がクラス CCTypeFactory を利用するようになったことに注意してください。 CCTypeFactory の目的は、 2 つのオブジェクトが同じタイプ番号を使用することを防ぎ、プラグインがユニークなタイプを持つとコントロール・センターに見

なされるようにすることです。新しいフォルダーは、いずれかの組み込み CC オブジェクト・タイプではなく、新しいタイプであるため、作成できる他の新しいタイプの番号と競合せず、なおかつ組み込みタイプの番号と競合しない、新しいタイプの番号が必要です。

getIcon メソッドは、オープンしたフォルダーかクローズしたフォルダーかを通知する iconState 用のパラメーターを入力として使用します。それによって、上記のように、アイコンを状態に対応させることができます。

データベースが選択された場合、ツリー内だけでなく、詳細ビューにフォルダーを表示するためには、 getData は値がプラグイン・オブジェクト自体である 1 つの列を戻す必要があります。 getData メソッドは、この 参照を、データ配列の最初のエレメントに割り当てます。これにより、アイコンと名前の両方を、詳細ビューの同じ列に表示できます。コントロール・センターは、 CCTableObject サブクラスが戻されることが分かると、 Example3Folder に対して getIcon および getName を呼び出せると判別します。

次のステップは、以下のようにして、CCTreeObject をインプリメントする CCDatabase クラスを作成し、その getChildren メソッドから Example3Folder のインスタンスを含む CCTableObject 配列を戻すことです。

```
import java.util.*;

  class CCDatabase implements CCTreeObject {
    private String parentName = null;
    private Vector childVector;

    public CCDatabase() {
      childVector = new Vector();
      childVector.addElement(new Example3Folder());
    }

    public CCTableObject[] getChildren() {
      CCTableObject[] children = new CCTableObject[childVector.size()];
      childVector.copyInto(children);
      return children;
    }

    public void setParentName(String name)
    {
      parentName = name;
    }

    public String getName () { return null; }
    public boolean isEditable () { return false; }
    public boolean isConfigurable () { return false; }
    public void getData (Object[] data) { }
    public CCColumn[] getColumns () { return null; }
    public boolean isLeaf () { return false; }
    public int getType () { return UDB_DATABASE; }
    public Icon getIcon (int iconState) { return null; }
    public CCMenuAction[] getMenuActions () { return null; }
  }
```

関連概念:

- 430 ページの『サンプル・プラグインのコンパイルおよび実行』

関連タスク:

- 441 ページの『フォルダーの下へのサンプル・オブジェクトの追加』

- 442 ページの『プラグイン・ツリー・オブジェクトの属性の設定』
- 444 ページの『Create アクションの追加』
- 445 ページの『複数選択をサポートする Remove アクションの追加』
- 447 ページの『Alter アクションの追加』

フォルダーの下へのサンプル・オブジェクトの追加

手順:

最初のステップは、次のようにして、子オブジェクト用の CCObject インプリメンテーションを作成することです。

```
  class Example3Child implements CCTableObject {
    private String parentName = null;
    public String getName () { return null; }
    public boolean isEditable () { return false; }
    public boolean isConfigurable () { return false; }
    public void getData (Object[] data) { }
    public CCColumn[] getColumns () { return null; }
    public Icon getIcon (int iconState) { return null; }
    public CCMenuAction[] getMenuActions () { return null; }
    public void setParentName(String name)
    {
      parentName = name;
    }
    public int getType () { return CCTypeFactory.getTypeNumber
(this.getClass().getName()); }
  }
```

次に、これらの Exercise3Child オブジェクトの Vector を保持するよう、以下のようにして、Example3Folder を変更します。

```
  public class Example3Folder implements CCObject {
    private String parentName = null;
    private Vector childVector;
    ...

    public Example3Folder() {
      childVector = new Vector();
    }

    ...
    public CCTableObject[] getChildren () {
      CCTableObject[] children = new CCTableObject[childVector.size()];
      childVector.copyInto(children);
      return children;
    }
    public void setParentName(String name)
    {
      parentName = name;
    }
    ...
  }
```

簡単にするために、この例では、 getChildren が childVector というベクトルに保管されている子の配列を戻します。

実際のプラグインでは、getChildren が呼び出されたときに子を再構成します。これにより、リストが更新され、前回リストが表示されたときより後にコントロール・

センターの外部で作成または変更された、新規または更新済みの子オブジェクトが表示されます。子は、永続ストレージに記憶され、永続ストレージから読み取られるため、失われることはありません。

また、実際のプラグインでは、コントロール・センターのツリーでどのオブジェクトがこのオブジェクトの親になっているかによって、 getChildren から戻される子のリストは変化します。親に関する情報は parentName ストリングにあります。このストリングは、setParentName メソッドにコントロール・センター呼び出しを行うと戻されます。

注: この例の場合は、Database オブジェクトか、ツリー内でそれより高位にあるオブジェクトからコントロール・センターでリフレッシュを実行すると、Example3 Folder より下位にある子のリストが失われます。これは、リフレッシュが実行されたときに、コントロール・センターによって新しい Example3Folder が構成されるためです。ただし、このサンプル・コードが永続ストレージから子を読み取っていれば、子は失われません。ここでは、例を簡単にするために、それを行いません。

関連概念:

- 430 ページの『サンプル・プラグインのコンパイルおよび実行』

関連タスク:

- 439 ページの『ツリーに複数のオブジェクトを入れるためのフォルダーの追加』
- 442 ページの『プラグイン・ツリー・オブジェクトの属性の設定』
- 444 ページの『Create アクションの追加』
- 445 ページの『複数選択をサポートする Remove アクションの追加』
- 447 ページの『Alter アクションの追加』

プラグイン・ツリー・オブジェクトの属性の設定

手順:

ツリーをプラグイン・フォルダーに展開して選択する場合、詳細ペインには列が示されません。これは、getColumns の Example3Child インプリメンテーションが NULL を戻すためです。これを変更するには、まずいくつかの CCColumn インプリメンテーションを作成します。 2 つの列を作成します。これから先の例では、このうち 1 つの列の値をランタイムに変更する方法を示しますが、各オブジェクトには変更されない 1 つの列が必要だからです。変更しない列を「Name」と呼び、変更する列を「State」と呼ぶことにします。

```
class NameColumn implements CCColumn {
   getName() { return "Name"; }
   getColumnClass { return CCTableObject.class; }
}

class StateColumn implements CCColumn {
   getName() { return "State"; }
   getColumnClass { return String.class; }
}
```

サポートされるクラス・タイプには、Java プリミティブと同等のクラス (java.lang.Intger など)、 java.util.Date クラス、および CCTableObject クラスが含まれます。

Example3Child の getColumns メソッドを変更し、これら 2 つの列を組み込みます。

```
class Example3Child implements CCTableObject {
  ...
  public CCColumn[] getColumns () {
     return new CCColumn[] { new NameColumn(),
                             new StateColumn() };
  }
  ...
}
```

さらに、親を変更して、同じ列を組み込みます。

```
class Example3Folder implements CCTableObject {
  ...
  public CCColumn[] getColumns () {
     return new CCColumn[] { new NameColumn(),
                             new StateColumn() };
  }
  ...
}
```

ここで、詳細ビューの行ごとに、表示される値を設定する必要があります。これを行うには、getData に渡された Object 配列のエレメントを設定します。各列のデータのクラスは、対応する列の getColumnClass によって戻されるクラスと一致していなければなりません。

```
class Example3Child implements CCTableObject {
  ...
  private String name;
  private String state;

  public Exampe3Child(String name, String state) {
    this.name = name;
    this.state = state;
  }
  ...
  public void getData (Object[] data) {
     data[0] = this;
    data[1] = state;
  }
  ...
}
```

この場合、クラス CCTableObject であった最初の列がこの値を持つことになります。これにより、コントロール・センターは、getName で戻されたテキストと、getIcon で戻されたアイコンの両方を表示できるようになります。それで、次のステップは、これらをインプリメントすることです。ツールバー・ボタンには、例 1 で使用したのと同じリフレッシュ・アイコンを使用します。

```
class Example3Child implements CCTableObject {
  ...
  public String getName () {
    return name;
  }
  public Icon getIcon () {
return CommonImageRepository.getScaledIcon(CommonImageRepository.WC_NV_
```

```
REFRESH);
    }
    ...
  }
```

これまでの作業の結果を確認するため、次の練習で除去できるような子オブジェクトの例を作成できます。 childVector が構成されたら、Example3Child のインスタンスを Example3Folder に追加します。

```
  public class Example3Folder implements CCTreeObject {
    ...
    public Example3Folder() {
      childVector = new Vector();
      childVector.addElement(new Example3Child("Plugin1", "State1"));
    }
    ...
  }
```

関連概念:

- 430 ページの『サンプル・プラグインのコンパイルおよび実行』

関連タスク:

- 439 ページの『ツリーに複数のオブジェクトを入れるためのフォルダーの追加』
- 441 ページの『フォルダーの下へのサンプル・オブジェクトの追加』
- 444 ページの『Create アクションの追加』
- 447 ページの『Alter アクションの追加』

Create アクションの追加

手順:

ユーザーがランタイムにフォルダーの下でオブジェクトを作成できるようにするため、Vector を更新し、クラスを Observable にし、さらにユーザーがイベントを生成するときに notifyObservers を呼び出す必要があります。コントロール・センターは、それ自体を、Observable である CCTableObject の Observer として自動的に登録します。

まず、子オブジェクトを子の Vector に追加するメソッドを Example3Folder に追加します。

```
  public class Example3Folder implements CCTreeObject, Observable {
    ...
    public void addChild(Example3Child child) {
      childVector.addElement(child);
      setChanged();
      notifyObservers(new CCObjectCollectionEvent(this,
                      CCObjectCollectionEvent.OBJECT_ADDED,
                      child));
    }
    ...
  }
```

上記のコードでは、CCObjectCollectionEvent という新しいクラスを、notifyObservers への引き数として使用しています。 CCObjectCollectionEvent は、コントロール・センター・ツリーのフォルダーなど、 CCObject の集合での変更を表すイベントです。コントロール・センターは、Observable にされるすべての

CCObject を監視し、ツリーおよび詳細ビューを更新することによって CCObjectCollectionEvent に応答します。イベントには、追加、除去、および変更の 3 つのタイプがあります。

CCObjectCollectionEvent は、3 つの引き数を使用します。最初の引き数は、イベントを生成したオブジェクトです。 2 番目の引き数は、イベントのタイプで、OBJECT_ADDED、OBJECT_ALTERED、または OBJECT_REMOVED のいずれかにできます。最後の引き数は、作成する新しいオブジェクトです。

次に、メニュー項目をフォルダーに追加して、ユーザーが新しい addChild メソッドへの呼び出しをトリガーできるようにします。

```
class CreateAction implements CCMenuAction {
  private int pluginNumber = 0;
  public String getMenuText () { return "Create"; }

  public void actionPerformed (ActionEvent e) {
Example3Folder folder = (Example3Folder)((Vector)e.getSource()).elementAt(0);
    folder.addChild(new Example3Child("Plugin " + ++pluginNumber, "State1"));
  }
}
```

ActionEvent には、必ず、アクションが呼び出されたオブジェクトすべての Vector が含まれます。このアクションは Example3Folder に対してのみ呼び出され、1 つのフォルダーしか存在できないため、 Vector の最初のオブジェクトをキャストし、それに対して addChild を呼び出すだけです。

最後のステップは、メニュー・アクションをフォルダーに追加することです。これで、以前に追加されたサンプル・オブジェクトを除去できるようになります。

```
public class Example3Folder extends Observable implements CCTreeObject {
  private CCMenuAction[] menuActions =
     new CCMenuAction[] { new CreateChildAction(); }
  ...
  public Example3Folder() {
    childVector = new Vector();
  }
  ...
  public CCMenuAction[] getMenuActions () {
    retun menuActions;
  }
  ...
}
```

関連概念:

- 430 ページの『サンプル・プラグインのコンパイルおよび実行』

関連タスク:

- 439 ページの『ツリーに複数のオブジェクトを入れるためのフォルダーの追加』
- 441 ページの『フォルダーの下へのサンプル・オブジェクトの追加』
- 442 ページの『プラグイン・ツリー・オブジェクトの属性の設定』
- 445 ページの『複数選択をサポートする Remove アクションの追加』
- 447 ページの『Alter アクションの追加』

複数選択をサポートする Remove アクションの追加

手順:

ここまでで、ユーザーは必要なだけのプラグインのインスタンスを作成できるようになり、それと同時にそれらのインスタンスを削除する権限も与えられるようになりました。まず、子を除去するメソッドを Example3Folder に追加し、コントロール・センターに通知します。

```
public class Example3Folder extends Observable implements CCTreeObject {

  public void removeChild(Example3Child child) {
    childVector.removeElement(child);
    setChanged();
    notifyObservers(new CCObjectCollectionEvent(this,
                    CCObjectCollectionEvent.OBJECT_REMOVED,
                    child));
  }
}
```

次のステップは、メニュー・アクションを Example3Child に追加することです。ユーザーが複数のオブジェクトを同時に除去できるように、この CCMenuAction インプリメントを MultiSelectable にします。このアクションのソースは Example3Folder ではなく Example3Child オブジェクトの Vector となるため、 Example3Folder を (コンストラクター内で渡すなどの) 別の方法でメニュー・アクションに渡す必要があります。

```
class RemoveAction implements CCMenuAction, MultiSelectable {
  private Example3Folder folder;

  public RemoveAction(Example3Folder folder) {
     this.folder = folder;
  }

  public String getMenuText () { return "Remove"; }

  public int getSelectionMode () { return MultiSelectable.MULTI_HANDLE_ONE; }

  public void actionPerformed (ActionEvent e) {
    Vector childrenVector = (Vector)e.getSource();
    for (int i = 0; i < childrenVector.size(); i++) {
      folder.removeChild((Example3Child)childrenVector.elementAt(i));
    }
  }
}
```

MultiSelectable をインプリメントするときには、getSelectionMode をインプリメントする必要があります。この場合、MULTI_HANDLE_ONE を戻すようになります。これは、複数のオブジェクトが選択されて、すべての選択済みオブジェクトのために actionPerformed メソッドが 1 回呼び出される場合でも、このメニュー項目がメニューに表示されることを意味します。

ここで、新しいメニュー・アクションを Example3Child に追加します。これを行うには、新しいパラメーターを Example3Child コンストラクターに追加し、フォルダーに渡します。

```
class Example3Child implements CCTableObject {
  ...
  private CCMenuAction[] menuActions;

  public Example3Child(Example3Folder folder, String name, String state) {
    ...
    menuActions = new CCMenuAction[] { new RemoveAction(folder) };
  }
  ...
```

```
    public CCMenuAction[] getMenuActions () {
      return menuActions;
    }
  }
```

新しいコンストラクターを使用するには、必ず CreateAction を変更してください。

```
  class CreateAction implements CCMenuAction {
    ...
    public void actionPerformed (ActionEvent e) {
      ...
 folder.addChild(new Example3Child(folder, "Plugin " + ++pluginNumber,
"State 1"));
    }
  }
```

関連概念:

- 430 ページの『サンプル・プラグインのコンパイルおよび実行』

関連タスク:

- 439 ページの『ツリーに複数のオブジェクトを入れるためのフォルダーの追加』
- 441 ページの『フォルダーの下へのサンプル・オブジェクトの追加』
- 442 ページの『プラグイン・ツリー・オブジェクトの属性の設定』
- 444 ページの『Create アクションの追加』
- 447 ページの『Alter アクションの追加』

Alter アクションの追加

手順:

プラグインに関連してコントロール・センターが listen する最後のタイプのイベントは、 OBJECT_ALTERED イベントです。この機能をこの例で実演できるように、前の例で ″State″ 列を作成しました。 Alter アクションが呼び出されると、状態値を増分することになります。

最初のステップは、状態を変更するメソッドを作成することですが、今回は、フォルダーではなく Example3Child 上で行います。この場合、最初の引き数と 3 番目の引き数はどちらも Example3Child です。必ず Observable を拡張してください。

```
  class Example3Child extends Observable implements CCTableObject {
    ...
    public void setState(String state) {
      this.state = state;
      setChanged();
      notifyObservers(new CCObjectCollectionEvent(this,
                          CCObjectCollectionEvent.OBJECT_ALTERED, this));
    }
    ...
  }
```

次に、Alter のメニュー・アクションを作成し、それを Example3Child の CCMenuAction 配列へ追加します。 AlterAction クラスはまた、ユーザーがコントロール・センターで Example3Child オブジェクトをダブルクリックするときに呼び出されるデフォルトのアクションとして、Alter を定義するために、CCDefaultMenuAction インターフェースをインプリメントします。

```
class AlterAction implements CCMenuAction, CCDefaultMenuAction {
  private int stateNumber = 1;
  public String getMenuText () { return "Alter"; }

  public void actionPerformed (ActionEvent e) {
    ((Example3Child)((Vector)e.getSource()).elementAt(0)).setState("State "
        + ++stateNumber);
  }
}

class Example3Child implements CCTableObject {
  ...
  public Example3Child(Example3Folder folder, String name, String state) {
    ...
    menuActions = new CCMenuAction[] { new AlterAction(),
                                       new RemoveAction(folder) };
  }
  ...
}
```

関連概念:

- 430 ページの『サンプル・プラグインのコンパイルおよび実行』

関連タスク:

- 439 ページの『ツリーに複数のオブジェクトを入れるためのフォルダーの追加』
- 441 ページの『フォルダーの下へのサンプル・オブジェクトの追加』
- 442 ページの『プラグイン・ツリー・オブジェクトの属性の設定』
- 444 ページの『Create アクションの追加』
- 445 ページの『複数選択をサポートする Remove アクションの追加』

isConfigurable() を使用して構成機能を使用不可にする

手順:

タイプ UDB_DATABASE または UDB_INSTANCE の CCObject の isConfigurable メソッドに false の値を戻すと、それぞれ「データベース (Database)」および「インスタンス (Instance)」ポップアップ・メニューから、「構成 (Configure)」メニュー項目が除去されます。

関連概念:

- 430 ページの『サンプル・プラグインのコンパイルおよび実行』

isEditable() を使用してオブジェクト変更機能を使用不可にする

手順:

Alter アクションをサポートするすべてのコントロール・センター・オブジェクトでは、そのオブジェクト用のプラグインを作成し、isEditable メソッドから false を戻すことによって、そのアクションを除去することができます。また、あるオブジェクトの getName メソッドで戻された値と一致する完全修飾名を持つオブジェクトの場合にのみ、 Alter アクションが除去されることを指定することも可能です。

別の機能は、Browse アクションの追加です。これは、UDB 表、ビュー、および索引に対してのみ行うことができます。この Browse アクションは、ファイルを

BrowseTable、BrowseView、または BrowseIndex という db2plug.zip に置くことによって追加されます。これらのファイルは空の場合がありますが、名前は、BrowseTable、 BrowseView、または BrowseIndex でなければなりません。「ブラウズ (Browse)」ボタンを追加することは、特定の名前が付けられたオブジェクトに制限できます。

関連概念:

- 430 ページの『サンプル・プラグインのコンパイルおよび実行』

hasConfigurationDefaults() を使用して構成ダイアログのデフォルト・ボタンを使用不可にする

手順:

UDB データベースおよびインスタンスの構成ダイアログには、値を DB2 デフォルトに戻す設定をするためのボタンがあります。独自のデフォルトを設定した場合、ユーザーが間違ってこれらのボタンを押すことを防ぐために、プラグインを使用してボタンを使用不可にすることができます。 CCObject をインプリメントするオブジェクトを作成し、そのタイプを UDB_DATABASE か UDB_INSTANCE に設定します。 Defaultable インターフェースもインプリメントしてください。このインターフェースには、hasConfigurationDefaults というメソッドが含まれています。このメソッドで false を戻すと、構成ダイアログのデフォルトのボタンはすべて使用不可になります。例では、UDB データベース構成のデフォルトのボタンが使用不可にされています。

関連概念:

- 430 ページの『サンプル・プラグインのコンパイルおよび実行』

付録 K. DB2 Universal Database 技術情報

DB2 資料とヘルプ

DB2® 技術情報は、以下のツールと方法を介して利用できます。

- DB2 インフォメーション・センター
  - トピック
  - DB2 ツールのヘルプ
  - サンプル・プログラム
  - チュートリアル
- ダウンロード可能な PDF ファイル、CD 上の PDF ファイル、および印刷された資料
  - ガイド
  - リファレンス・マニュアル
- コマンド行ヘルプ
  - コマンド・ヘルプ
  - メッセージ・ヘルプ
  - SQL 状態ヘルプ
- インストール済みソース・コード
  - サンプル・プログラム

ibm.com® にある技術資料、白書、 Redbooks™ その他の DB2 Universal Database™ 技術情報にオンラインでアクセスできます。 DB2 Information Management ソフトウェア・ライブラリー・サイト (www.ibm.com/software/data/pubs/) にアクセスしてください。

DB2 資料の更新

IBM® は、 DB2 インフォメーション・センターの資料のフィックスパックやその他の資料更新を定期的に発行しています。 DB2 インフォメーション・センター (http://publib.boulder.ibm.com/infocenter/db2help/) にアクセスすれば、常に最新の情報が掲載されます。 DB2 インフォメーション・センターをローカル・インストールしている場合、更新記事を表示するには、まず手動で更新をインストールしてください。新しい情報が発表されたときに資料を更新することにより、 *DB2 インフォメーション・センター CD* からインストールした情報を更新することができます。

インフォメーション・センターの方が、PDF 資料やハードコピー資料よりも頻繁に更新されます。 DB2 の最新の技術情報を入手するには、資料更新が発行されたときにそれをインストールするか、または www.ibm.com サイトの DB2 インフォメーション・センターにアクセスしてください。

関連概念:

- 「*コール・レベル・インターフェース ガイドおよびリファレンス 第 1 巻*」の『CLI サンプル・プログラム』

- 「*アプリケーション開発ガイド アプリケーションの構築および実行*」の『Java サンプル・プログラム』
- 452 ページの『DB2 インフォメーション・センター』

関連タスク:

- 473 ページの『DB2 ツールからコンテキスト・ヘルプを呼び出す』
- 463 ページの『コンピューターまたはイントラネット・サーバーへの DB2 インフォメーション・センターの更新インストール』
- 474 ページの『コマンド行プロセッサーからメッセージ・ヘルプを呼び出す』
- 474 ページの『コマンド行プロセッサーからコマンド・ヘルプを呼び出す』
- 475 ページの『コマンド行プロセッサーから SQL 状態ヘルプを呼び出す』

関連資料:

- 465 ページの『DB2 PDF 資料および印刷された資料』

DB2 インフォメーション・センター

DB2® インフォメーション・センターを使用すると、 DB2 Universal Database™、 DB2 Connect™、 DB2 Information Integrator および DB2 Query Patroller™ などの DB2 ファミリー製品を最大限に活用するのに必要なすべての情報にアクセスできます。また、DB2 インフォメーション・センターは、 DB2 の主な機能とコンポーネントに関する情報を提供します (レプリケーション、データウェアハウジング、および DB2 の種々の Extender など)。

Mozilla 1.0 以上または Microsoft® Internet Explorer 5.5 以上で表示する場合、 DB2 インフォメーション・センターには以下の機能があります。以下のいくつかの機能では、 JavaScript™ のサポートを使用可能にする必要があります:

柔軟なインストール・オプション
以下の中から、 ご使用の環境に最も適したオプションを使って DB2 資料を表示できます。

- 最新の資料を常に自動的に利用できるようにするには、 IBM® の Web サイト (http://publib.boulder.ibm.com/infocenter/db2help/) にある DB2 インフォメーション・センターからすべての資料に直接アクセスします。
- 更新処理を最小化し、イントラネット内のネットワーク・トラフィックだけに制限するには、イントラネット上の 1 つのサーバーに DB2 資料をインストールします。
- 柔軟性を改善し、ネットワーク接続への依存を軽減するには、個々のコンピューターに DB2 資料をインストールします。

検索 「**検索**」テキスト・フィールドに検索語を入力することにより、 DB2 インフォメーション・センターのすべてのトピックを検索できます。複数の語句を引用符で囲めば、完全一致を検索できます。また、ワイルドカード演算子 (*、?) とブール演算子 (AND、 NOT、OR) を使用して検索を絞り込むことができます。

タスク指向の目次
単一の目次の中から、DB2 資料のトピックを見付けることができます。目

次は、主に実行するタスクの種類に従って編成されていますが、そのほかに製品概要、特定のゴール (目的) の情報、参照情報、索引、および用語集も含まれます。

- 製品概要では、DB2 ファミリーで使用可能な製品間の関係、そうした各製品で提供される機能、および各製品の最新リリース情報について説明されています。
- インストール、管理および開発などのゴール・カテゴリーには、タスクを迅速に完了し、そのための背景情報をよく理解できるようにするトピックが含まれています。
- 「参照」トピックでは、その対象に関する詳細な情報 (ステートメントとコマンドの構文、メッセージ・ヘルプ、構成パラメーターなど) が説明されています。

現在のトピックを目次に表示する

現在のトピックが目次のどの部分に該当するかを表示するには、目次フレーム内の**「リフレッシュ/現在のトピックの表示 (Refresh/Show Current Topic)」**ボタンをクリックするか、コンテンツ・フレーム内の**「目次に表示 (Show in Table of Contents)」**ボタンをクリックします。幾つかのファイルで関連トピックへの複数のリンクをたどった場合、または検索結果からトピックにアクセスした場合には、この機能が役立ちます。

索引 索引から、すべての資料にアクセスすることができます。索引では、用語が 50 音順に編成されています。

用語集 用語集を見れば、DB2 資料で使われているさまざまな用語の定義を調べることができます。用語集では、用語が 50 音順に編成されています。

組み込まれているローカライズ情報

DB2 インフォメーション・センターは、ブラウザーで設定された言語でトピックを表示します。設定された言語のトピックが利用できない場合、DB2 インフォメーション・センターにはそのトピックの英語版が表示されます。

iSeries™ 技術情報については、 IBM eServer™ iSeries Information Center (www.ibm.com/eserver/iseries/infocenter/) を参照してください。

関連概念:

- 454 ページの『DB2 インフォメーション・センターのインストール・シナリオ』

関連タスク:

- 463 ページの『コンピューターまたはイントラネット・サーバーへの DB2 インフォメーション・センターの更新インストール』
- 464 ページの『DB2 インフォメーション・センターにおける特定の言語でのトピックの表示』
- 462 ページの『DB2 インフォメーション・センターの呼び出し』
- 456 ページの『DB2 セットアップ・ウィザードを使用した DB2 インフォメーション・センターのインストール (UNIX)』
- 459 ページの『DB2 セットアップ・ウィザードを使用した DB2 インフォメーション・センターのインストール (Windows)』

DB2 インフォメーション・センターのインストール・シナリオ

さまざまに異なる業務環境のもとでは、 DB2® 情報にどのようにアクセスするかの要件もそれぞれ異なります。 DB2 インフォメーション・センターにアクセスするには、 IBM® の Web サイト、サーバーまたは組織のネットワーク、あるいはコンピューターへのインストールという 3 つの方法が可能です。この 3 つのケースのいずれも、資料は DB2 インフォメーション・センター内に置かれます。インフォメーション・センターは、ブラウザーを使って表示できるように設計されたトピック・ベースの情報の Web サイトです。デフォルトでは、DB2 製品から、 IBM Web サイト上の DB2 インフォメーション・センターにアクセスします。これに対して、 イントラネット・サーバーまたはご自分のコンピューターから DB2 インフォメーション・センターにアクセスしたい場合、製品メディア・パック内にある DB2 インフォメーション・センター CD から DB2 インフォメーション・センターをインストールする必要があります。以下では、DB2 資料へのアクセス・オプションの要約、および 3 つのインストール・シナリオを示します。これを参考にして、お客様の業務環境で DB2 インフォメーション・センターにアクセスするにはどの方法が最適か、どのようなインストール上の問題に配慮する必要があるかを判別してください。

DB2 資料にアクセスするオプションの要約:

以下の表は、お客様の実際の業務環境で、 DB2 インフォメーション・センターの DB2 製品情報にアクセスする方法としてどんなオプションが推奨されるかを示します。

インターネット・アクセス	イントラネット・アクセス	推奨されるアクション
はい	はい	IBM Web サイト上の DB2 インフォメーション・センターへのアクセス、またはイントラネット・サーバーにインストール済みの DB2 インフォメーション・センターへのアクセス
はい	いいえ	IBM Web サイト上の DB2 インフォメーション・センターへのアクセス
いいえ	はい	イントラネット・サーバーにインストール済みの DB2 インフォメーション・センターへのアクセス
いいえ	いいえ	ローカル・コンピューター上の DB2 インフォメーション・センターへのアクセス

シナリオ: コンピューター上の DB2 インフォメーション・センターへのアクセス:

Tsu-Chen 氏は小さな町で工場を経営していますが、その町には、インターネット・アクセスを提供する地元のインターネット・サービス・プロバイダーがありません。彼は、在庫、製品オーダー、銀行口座情報、および営業経費を管理するために DB2 Universal Database™ を購入しました。 Tsu-Chen 氏は以前に DB2 製品を利用したことがないので、 DB2 の使用方法を習得するために、DB2 製品資料を参照する必要があります。

Tsu-Chen 氏は 標準インストール・オプションを使って DB2 Universal Database を自分のコンピューターにインストールした後、 DB2 資料にアクセスしようとします。しかし、開こうとしているページが見つからないというエラー・メッセージがブラウザーから通知されました。 Tsu-Chen 氏は DB2 製品のインストール・マニュアルを調べた結果、 DB2 資料を自分のコンピューター上で利用するには、 DB2 インフォメーション・センターをインストールしなければならないことに気付きます。そしてメディア・パックの中にあった *DB2 インフォメーション・センター CD* を見つけ出して、インストールしました。

これで、Tsu-Chen 氏はオペレーティング・システムのアプリケーション・ランチャーから DB2 インフォメーション・センターにアクセスできるようになり、より良い業務成果をあげるために DB2 製品を利用する方法を習得できます。

シナリオ: IBM Web サイト上の DB2 インフォメーション・センターへのアクセス:

Colin は、あるセミナー企業に所属する情報技術コンサルタントです。彼の専門はデータベース・テクノロジーおよび SQL で、 DB2 Universal Database を使って北米一帯の企業を対象にこれらの科目のセミナーを開催しています。 Colin のセミナーでは、教材として DB2 資料も使用されます。たとえば、SQL の講習コースでは、データベース照会の基本構文と拡張構文を教えるために SQL に関する DB2 資料が使用されます。

Colin が教えている企業の大半はインターネット・アクセスを配備しています。このような状況から判断して、Colin は、最新バージョンの DB2 Universal Database を自分のモバイル・コンピューターにインストールしたとき、 IBM Web サイト上の DB2 インフォメーション・センターにアクセスするよう構成しました。この構成によって、Colin はセミナーで教えるときに最新の DB2 資料にオンライン・アクセスすることができます。

しかし、時折、Colin は移動中にインターネット・アクセスを利用できないことがあります。これは問題となります。担任するセミナーの準備のために DB2 資料にアクセスする必要のある場合には、とくにそうです。このような事態が起きないようにするために、 Colin は自分のモバイル・コンピューターに DB2 インフォメーション・センターのコピーをインストールしました。

こうして、Colin は常に DB2 資料のコピーを自在に活用できるようになりました。**db2set** コマンドを使って自分のモバイル・コンピューターのレジストリー変数を簡単に構成し、どこにいるかに応じて、IBM Web サイトまたは自分のモバイル・コンピューターから DB2 インフォメーション・センターにアクセスできます。

シナリオ: イントラネット・サーバー上の DB2 インフォメーション・センターへのアクセス:

Eva は、生命保険会社のデータベース上級管理者です。彼女は管理業務の一環として、会社の UNIX® データベース・サーバーに最新バージョンの DB2 Universal Database をインストールおよび構成します。彼女の会社は最近、セキュリティー上の理由から、インターネット・アクセスをもはや業務で利用できないようにすると社員に通知しました。同社はネットワーク環境を装備しているため、 Eva は DB2 インフォメーション・センターのコピーをイントラネット・サーバー上にインスト

ールして、社内のデータウェアハウスを定期的に利用するすべての社員 (営業担当者、営業部長、および業務分析担当者) から DB2 資料へのアクセスを可能にすることにしました。

Eva は、応答ファイルを使って全社員のコンピューター上に最新バージョンの DB2 Universal Database をインストールするようデータベース・チームに指示します。その際、イントラネット・サーバーのホスト名とポート番号を使って DB2 インフォメーション・センターにアクセスできるよう、確実に各コンピューターを構成します。

しかし、Eva のチームの下級データベース管理者である Migual の誤解によって、数人の社員のコンピューター上で、イントラネット・サーバーの DB2 インフォメーション・センターにアクセスするよう DB2 Universal Database を構成する代わりに、 DB2 インフォメーション・センターのコピーをそれらのコンピューターにインストールしてしまいました。これを訂正するために、Eva は、 **db2set** コマンドを使ってこれらのコンピューター上の DB2 インフォメーション・センターのレジストリー変数 (ホスト名は DB2_DOCHOST、ポート番号は DB2_DOCPORT) を変更するよう Migual に指示しました。これで、ネットワーク上の適切なすべてのコンピューターが DB2 インフォメーション・センターにアクセスできるようになり、社員は DB2 に関する質問の答えを DB2 資料から見つけることができます。

関連概念:

- 452 ページの『DB2 インフォメーション・センター』

関連タスク:

- 463 ページの『コンピューターまたはイントラネット・サーバーへの DB2 インフォメーション・センターの更新インストール』
- 456 ページの『DB2 セットアップ・ウィザードを使用した DB2 インフォメーション・センターのインストール (UNIX)』
- 459 ページの『DB2 セットアップ・ウィザードを使用した DB2 インフォメーション・センターのインストール (Windows)』
- 『DB2 インフォメーション・センターへのアクセスのロケーションの設定: Common GUI help』

関連資料:

- 「コマンド・リファレンス」の『db2set - DB2 プロファイル・レジストリー・コマンド』

DB2 セットアップ・ウィザードを使用した DB2 インフォメーション・センターのインストール (UNIX)

DB2 製品資料にアクセスする方法として、 IBM Web サイト、イントラネット・サーバー、またはコンピューターにインストールしたバージョンの 3 つがあります。デフォルトでは、DB2 製品は IBM Web サイト上の DB2 資料にアクセスします。イントラネット・サーバーまたはコンピューター上の DB2 資料にアクセスしたい場合には、 *DB2 インフォメーション・センター CD* から資料をインストールする必要があります。 DB2 セットアップ・ウィザードを使用すれば、インストール設

定を定義し、 UNIX オペレーティング・システムを使用するコンピューターに DB2 インフォメーション・センターをインストールできます。

前提条件:

このセクションでは、UNIX コンピューターに DB2 インフォメーション・センターをインストールするためのハードウェア、オペレーティング・システム、ソフトウェア、および通信の諸要件を一覧で示します。

- **ハードウェア要件**

  以下のいずれかのプロセッサーが必要です。

  - PowerPC (AIX)
  - HP 9000 (HP-UX)
  - Intel 32 ビット (Linux)
  - Solaris UltraSPARC コンピューター (Solaris オペレーティング環境)

- **オペレーティング・システム要件**

  以下のいずれかのオペレーティング・システムが必要です。

  - IBM AIX 5.1 (PowerPC 上)
  - HP-UX 11i (HP 9000 上)
  - Red Hat Linux 8.0 (Intel 32 ビット上)
  - SuSE Linux 8.1 (Intel 32 ビット上)
  - Sun Solaris バージョン 8 (Solaris オペレーティング環境の UltraSPARC コンピューター上)

  注: DB2 インフォメーション・センターは、 DB2 クライアントをサポートする UNIX オペレーティング・システム上で稼動します。このため、IBM Web サイトから DB2 インフォメーション・センターにアクセスするか、イントラネット・サーバーに DB2 インフォメーション・センターをインストールしてそれにアクセスすることをお勧めします。

- **ソフトウェア要件**
  - 以下のブラウザーがサポートされています。
    - Mozilla バージョン 1.0 以上
- DB2 セットアップ・ウィザードは、グラフィック・インストーラーです。ご使用のマシンで DB2 セットアップ・ウィザードのグラフィカル・ユーザー・インターフェースを表示可能にする X Window システム・ソフトウェアをインプリメントする必要があります。 DB2 セットアップ・ウィザードを実行する前に、ディスプレイを正しくエクスポートしたことを確認してください。たとえば、コマンド・プロンプトで

  ```
  export DISPLAY=9.26.163.144:0.
  ```

  というコマンドを入力します。

- **通信要件**
  - TCP/IP

手順:

DB2 セットアップ・ウィザードを使用して DB2 インフォメーション・センターをインストールするには、以下のようにします。

1. システムにログオンします。
2. DB2 インフォメーション・センター製品 CD を挿入してシステムにマウントします。
3. 次のコマンドを入力して、CD がマウントされているディレクトリーに移動します。

   ```
   cd /cd
   ```

   /cd は、CD のマウント・ポイントを表します。
4. **./db2setup** コマンドを入力して、 DB2 セットアップ・ウィザードを開始します。
5. IBM DB2 セットアップ・ランチパッドが開きます。 DB2 インフォメーション・センターのインストールに直接進むには、**「製品のインストール」**をクリックします。残りのステップについて説明しているオンライン・ヘルプを利用できます。オンライン・ヘルプを呼び出すには、**「ヘルプ」**をクリックします。**「キャンセル」**をクリックすれば、いつでもインストールを終了できます。
6. **「インストールしたい製品を選択します」**ページでは、**「次へ」**をクリックします。
7. **「DB2 セットアップ・ウィザードにようこそ (Welcome to the DB2 Setup wizard)」**ページで、**「次へ」**をクリックします。 DB2 セットアップ・ウィザードは、プログラムのセットアップ操作を案内します。
8. インストールを続行するには、使用許諾条件に同意する必要があります。**「ご使用条件」**ページで、**「ご使用条件に同意します (I accept the terms in the license agreement)」** をクリックして、**「次へ」**をクリックします。
9. **「インストール・アクションの選択」**で、**「このコンピューターに DB2 インフォメーション・センターをインストールする (Install DB2 Information Center on this computer)」**を選択します。応答ファイルを使用して、このコンピューターまたは他のコンピューターに DB2 インフォメーション・センターをあとでインストールしたい場合には、**「設定を応答ファイルに保管する」**を選択します。**「次へ」**をクリックします。
10. **「インストールする言語の選択」**ページでは、 DB2 インフォメーション・センターをインストールする言語を選択します。**「次へ」**をクリックします。
11. **「DB2 インフォメーション・センター・ポートの指定」**ページでは、 DB2 インフォメーション・センターへの着信通信を構成します。**「次へ」**をクリックしてインストールを続けます。
12. **「ファイルのコピーの開始」**ページでは、インストールの選択項目を確認します。設定を変更するには、**「戻る」**をクリックします。**「インストール」**をクリックすると、 DB2 インフォメーション・センターのファイルがコンピューターにコピーされます。

このほか、応答ファイルを使って DB2 インフォメーション・センターをインストールすることもできます。

インストール・ログ db2setup.his、 db2setup.log、および db2setup.err は、デフォルトでは /tmp ディレクトリーに置かれます。

db2setup.log ファイルは、エラーも含めた DB2 製品のインストール情報をすべてキャプチャーします。 db2setup.his ファイルは、コンピューター上の DB2 製品インストール内容をすべて記録します。 DB2 は、db2setup.log ファイルを db2setup.his に付加します。 db2setup.err ファイルは、 Java から戻されるすべてのエラー出力 (例外やトラップの情報など) をキャプチャーします。

インストールが完了したら、ご使用の UNIX オペレーティング・システムに応じて、 DB2 は以下のいずれかのディレクトリーにインストールされます。

- AIX: /usr/opt/db2_08_01
- HP-UX: /opt/IBM/db2/V8.1
- Linux: /opt/IBM/db2/V8.1
- Solaris オペレーティング環境: /opt/IBM/db2/V8.1

関連概念:

- 452 ページの『DB2 インフォメーション・センター』
- 454 ページの『DB2 インフォメーション・センターのインストール・シナリオ』

関連タスク:

- 「*インストールおよび構成 補足*」の『応答ファイルによる DB2 のインストール (UNIX)』
- 463 ページの『コンピューターまたはイントラネット・サーバーへの DB2 インフォメーション・センターの更新インストール』
- 464 ページの『DB2 インフォメーション・センターにおける特定の言語でのトピックの表示』
- 462 ページの『DB2 インフォメーション・センターの呼び出し』
- 459 ページの『DB2 セットアップ・ウィザードを使用した DB2 インフォメーション・センターのインストール (Windows)』

DB2 セットアップ・ウィザードを使用した DB2 インフォメーション・センターのインストール (Windows)

DB2 製品資料にアクセスする方法として、 IBM Web サイト、イントラネット・サーバー、またはコンピューターにインストールしたバージョンの 3 つがあります。デフォルトでは、DB2 製品は IBM Web サイト上の DB2 資料にアクセスします。イントラネット・サーバーまたはコンピューター上の DB2 資料にアクセスしたい場合には、 *DB2 インフォメーション・センター CD* から DB2 資料をインストールする必要があります。 DB2 セットアップ・ウィザードを使用すれば、インストール設定を定義し、 Windows オペレーティング・システムを使用するコンピューターに DB2 インフォメーション・センターをインストールできます。

前提条件:

このセクションでは、Windows に DB2 インフォメーション・センターをインストールするためのハードウェア、オペレーティング・システム、ソフトウェア、および通信の諸要件を一覧で示します。

- **ハードウェア要件**

  以下のいずれかのプロセッサーが必要です。

  – 32 ビット・コンピューター: Pentium または Pentium 互換の CPU

- **オペレーティング・システム要件**

  以下のいずれかのオペレーティング・システムが必要です。

  – Windows 2000

  – Windows XP

  注: DB2 インフォメーション・センターは、 DB2 クライアントをサポートする Windows オペレーティング・システム上で稼動します。このため、IBM Web サイトの DB2 インフォメーション・センターにアクセスするか、イントラネット・サーバーに DB2 インフォメーション・センターをインストールしてそれにアクセスすることをお勧めします。

- **ソフトウェア要件**

  – 以下のブラウザーがサポートされています。

    - Mozilla 1.0 以上

    - Internet Explorer バージョン 5.5 または 6.0 (Windows XP の場合はバージョン 6.0)

- **通信要件**

  – TCP/IP

制約事項:

- DB2 インフォメーション・センターをインストールするには、管理権限をもつアカウントが必要です。

手順:

DB2 セットアップ・ウィザードを使用して DB2 インフォメーション・センターをインストールするには、以下のようにします。

1. DB2 インフォメーション・センターのインストールで定義したアカウントで、システムにログオンします。
2. CD をドライブに挿入します。自動実行機能が使用可能になっていれば、IBM DB2 セットアップ・ランチパッドが起動します。
3. DB2 セットアップ・ウィザードは、システム言語を判別して、その言語用のセットアップ・プログラムを立ち上げます。英語以外の言語でセットアップ・プログラムを実行したい場合、またはセットアップ・プログラムの自動始動が失敗した場合には、 DB2 セットアップ・ウィザードを手動で開始できます。

   次のようにして、DB2 セットアップ・ウィザードを手動で開始します。

   a. **「スタート」**をクリックし、**「ファイル名を指定して実行」**を選択します。

   b. **「開く」**フィールドで、以下のコマンドを入力します。

      ```
      x:¥setup.exe /i 2-letter language identifier
      ```

ここで、*x:* は CD ドライブ、 *2-letter language identifier* (2 文字の言語識別子) はセットアップ・プログラムを実行する言語を表します。

c. **「OK」**をクリックします。

4. IBM DB2 セットアップ・ランチパッドが開きます。 DB2 インフォメーション・センターのインストールに直接進むには、**「製品のインストール」**をクリックします。残りのステップについて説明しているオンライン・ヘルプを利用できます。オンライン・ヘルプを呼び出すには、**「ヘルプ」**をクリックします。**「キャンセル」**をクリックすれば、いつでもインストールを終了できます。
5. **「インストールしたい製品を選択します」**ページでは、**「次へ」**をクリックします。
6. **「DB2 セットアップ・ウィザードにようこそ (Welcome to the DB2 Setup wizard)」**ページで、**「次へ」**をクリックします。 DB2 セットアップ・ウィザードは、プログラムのセットアップ操作を案内します。
7. インストールを続行するには、使用許諾条件に同意する必要があります。**「ご使用条件」**ページで、**「ご使用条件に同意します (I accept the terms in the license agreement)」** をクリックして、**「次へ」**をクリックします。
8. **「インストール・アクションの選択」**で、**「このコンピューターに DB2 インフォメーション・センターをインストールする (Install DB2 Information Center on this computer)」**を選択します。応答ファイルを使用して、このコンピューターまたは他のコンピューターに DB2 インフォメーション・センターをあとでインストールしたい場合には、**「設定を応答ファイルに保管する」**を選択します。**「次へ」**をクリックします。
9. **「インストールする言語の選択」**ページでは、 DB2 インフォメーション・センターをインストールする言語を選択します。**「次へ」**をクリックします。
10. **「DB2 インフォメーション・センター・ポートの指定」**ページでは、 DB2 インフォメーション・センターへの着信通信を構成します。**「次へ」**をクリックしてインストールを続けます。
11. **「ファイルのコピーの開始」**ページでは、インストールの選択項目を確認します。設定を変更するには、**「戻る」**をクリックします。**「インストール」**をクリックすると、 DB2 インフォメーション・センターのファイルがコンピューターにコピーされます。

応答ファイルを使って DB2 インフォメーション・センターをインストールすることができます。また、**db2rspgn** コマンドを使って、既存のインストール内容に基づく応答ファイルを生成することもできます。

インストール時に検出されるエラーの詳細については、 「マイ ドキュメント」¥DB2LOG¥ ディレクトリー内の db2.log ファイルと db2wi.log ファイルを参照してください。 「マイ ドキュメント」 ディレクトリーの場所は、ご使用のコンピューターの設定によって異なります。

db2wi.log ファイルは、 DB2 の最新のインストール情報をキャプチャーします。db2.log は、DB2 製品のインストールの履歴をキャプチャーします。

関連概念:

• 452 ページの『DB2 インフォメーション・センター』

- 454 ページの『DB2 インフォメーション・センターのインストール・シナリオ』

関連タスク:

- 「*インストールおよび構成 補足*」の『応答ファイルによる DB2 製品のインストール (Windows)』
- 463 ページの『コンピューターまたはイントラネット・サーバーへの DB2 インフォメーション・センターの更新インストール』
- 464 ページの『DB2 インフォメーション・センターにおける特定の言語でのトピックの表示』
- 462 ページの『DB2 インフォメーション・センターの呼び出し』
- 456 ページの『DB2 セットアップ・ウィザードを使用した DB2 インフォメーション・センターのインストール (UNIX)』

関連資料:

- 「コマンド・リファレンス」の『db2rspgn - 応答ファイル生成プログラム・コマンド』

DB2 インフォメーション・センターの呼び出し

DB2 インフォメーション・センターは、Linux、UNIX、および Windows オペレーティング・システム用の DB2 製品 (DB2 Universal Database、 DB2 Connect、DB2 Information Integrator、 DB2 Query Patroller など) を使用するために必要なすべての情報を提供します。

DB2 インフォメーション・センターは、以下の場所から呼び出すことができます。

- DB2 UDB クライアントまたはサーバーがインストールされているコンピューター
- DB2 インフォメーション・センターがインストールされているイントラネット・サーバーまたはローカル・コンピューター
- IBM の Web サイト

前提条件:

DB2 インフォメーション・センターを呼び出すための要件は、以下のとおりです。

- *オプショナル*: 希望する言語でトピックを表示するようブラウザーを構成する
- *オプショナル*: コンピューターまたはイントラネット・サーバーにインストール済みの DB2 インフォメーション・センターを使用するよう DB2 クライアントを構成する

手順:

DB2 UDB クライアントまたはサーバーがインストールされているコンピューターから DB2 インフォメーション・センターを呼び出すには、以下のようにします。

- (Windows オペレーティング・システムの)「スタート」メニューから: **「スタート」→「プログラム」→「IBM DB2」→「情報」→「インフォメーション・センター」**をクリックします。
- コマンド行プロンプトから:
  - Linux および UNIX オペレーティング・システムの場合、 **db2icdocs** コマンドを発行します。

– Windows オペレーティング・システムの場合、 **db2icdocs.exe** コマンドを発行します。

イントラネット・サーバーまたはローカル・コンピューターにインストール済みの DB2 インフォメーション・センターを Web ブラウザーで開くには、以下のようにします。

- Web ページ http://<host-name>:<port-number>/ を開きます (<host-name> はホスト名、 <port-number> は DB2 インフォメーション・センターを利用可能なポート番号)。

IBM Web サイトにある DB2 インフォメーション・センターを Web ブラウザーで開くには、以下のようにします。

- Web ページ publib.boulder.ibm.com/infocenter/db2help/ を開きます。

関連概念:

- 452 ページの『DB2 インフォメーション・センター』
- 454 ページの『DB2 インフォメーション・センターのインストール・シナリオ』

関連タスク:

- 473 ページの『DB2 ツールからコンテキスト・ヘルプを呼び出す』
- 463 ページの『コンピューターまたはイントラネット・サーバーへの DB2 インフォメーション・センターの更新インストール』
- 474 ページの『コマンド行プロセッサーからコマンド・ヘルプを呼び出す』
- 『DB2 インフォメーション・センターへのアクセスのロケーションの設定: Common GUI help』

関連資料:

- 「コマンド・リファレンス」の『HELP コマンド』

コンピューターまたはイントラネット・サーバーへの DB2 インフォメーション・センターの更新インストール

http://publib.boulder.ibm.com/infocenter/db2help/ から利用できる DB2 インフォメーション・センターは、資料の新規追加または変更によって定期的に更新されます。さらに、更新された DB2 インフォメーション・センターをコンピューターまたはイントラネット・サーバーにダウンロードしてインストールできる場合もあります。DB2 インフォメーション・センターを更新しても、 DB2 クライアント製品またはサーバー製品は更新されません。

前提条件:

インターネットに接続されたコンピューターへのアクセスが必要です。

手順:

DB2 インフォメーション・センターの更新をコンピューターまたはイントラネット・サーバーにインストールするには、以下のようにします。

1. IBM の Web サイト (http://publib.boulder.ibm.com/infocenter/db2help/) にある DB2 インフォメーション・センターを開きます。

2. 「DB2 インフォメーション・センターにようこそ」ページの見出し「サービスおよびサポート」の「ダウンロード」セクションで、**「DB2 資料」**リンクをクリックします。
3. 最新のドキュメンテーション・イメージのレベルと、インストール済みのドキュメンテーション・レベルを比較して、 DB2 インフォメーション・センターを更新する必要があるかどうかを確認します。「DB2 インフォメーション・センターにようこそ」ページに、インストール済みのドキュメンテーションのレベルがリストされます。
4. より新しいバージョンの DB2 インフォメーション・センターが存在する場合、ご使用のオペレーティング・システムに対応する最新の *DB2 インフォメーション・センター・イメージをダウンロードします。*
5. 最新の *DB2 インフォメーション・センター・イメージをインストールするには、* Web ページの指示に従ってください。

関連概念:

- 454 ページの『DB2 インフォメーション・センターのインストール・シナリオ』

関連タスク:

- 462 ページの『DB2 インフォメーション・センターの呼び出し』
- 456 ページの『DB2 セットアップ・ウィザードを使用した DB2 インフォメーション・センターのインストール (UNIX)』
- 459 ページの『DB2 セットアップ・ウィザードを使用した DB2 インフォメーション・センターのインストール (Windows)』

DB2 インフォメーション・センターにおける特定の言語でのトピックの表示

DB2 インフォメーション・センターでは、ブラウザーの設定で指定した言語でのトピックの表示が試みられます。トピックがその指定言語に翻訳されていない場合は、 DB2 インフォメーション・センターでは英語でトピックが表示されます。

手順:

Internet Explorer Web ブラウザーで、指定どおりの言語でトピックを表示するには、以下のようにします。

1. Internet Explorer の**「ツール」**—> **「インターネット オプション」**—> **「言語...」**ボタンをクリックします。「言語の優先順位」ウィンドウがオープンします。
2. 該当する言語が、言語リストの先頭の項目に指定されていることを確認します。
   - リストに新しい言語を追加するには、**「追加...」**ボタンをクリックします。

     注: 言語を追加しても、特定の言語でトピックを表示するのに必要なフォントがコンピューターに備えられているとはかぎりません。
   - リストの先頭に新しい言語を移動するには、その言語を選択してから、その言語が言語リストに先頭に行くまで**「上へ」**ボタンをクリックします。
3. 使いたい言語で DB2 インフォメーション・センターを表示するには、ページをリフレッシュします。

Mozilla Web ブラウザーの場合に、使いたい言語でトピックを表示するには、以下のようにします。

1. Mozilla の**「編集」**—> **「設定」**—> **「言語」**ボタンをクリックします。「設定」ウィンドウに「言語」パネルが表示されます。
2. 該当する言語が、言語リストの先頭の項目に指定されていることを確認します。
   - リストに新しい言語を追加するには、**「追加...」**ボタンをクリックしてから、「言語を追加」ウィンドウで言語を選択します。
   - リストの先頭に新しい言語を移動するには、その言語を選択してから、その言語が言語リストに先頭に行くまで**「上に移動」**ボタンをクリックします。
3. 使いたい言語で DB2 インフォメーション・センターを表示するには、ページをリフレッシュします。

関連概念:

- 452 ページの『DB2 インフォメーション・センター』

DB2 PDF 資料および印刷された資料

以下の表は、正式な資料名、資料番号、および PDF ファイル名を示しています。ハードコピー版の資料を注文するには、正式な資料名を知っておく必要があります。 PDF ファイルを印刷するには、PDF ファイル名を知っておく必要があります。

DB2 資料は、以下のカテゴリーに分類されています。

- DB2 中核情報
- 管理情報
- アプリケーション開発情報
- ビジネス・インテリジェンス情報
- DB2 Connect 情報
- 入門情報
- チュートリアル情報
- オプショナル・コンポーネント情報
- リリース・ノート

以下の表は、DB2 ライブラリー内の各資料について、その資料のハードコピー版を注文したり、PDF 版を印刷または表示したりするのに必要な情報を示しています。DB2 ライブラリー内の各資料に関する詳細な説明については、www.ibm.com/shop/publications/order にある IBM Publications Center にアクセスしてください。

DB2 の基本情報

こうした資料の情報は、すべての DB2 ユーザーに基本的なもので、プログラマーおよびデータベース管理者にとって役立つ情報であるとともに、DB2 Connect、DB2 Warehouse Manager、または他の DB2 製品を使用するユーザーにとっても役立つ内容です。

表 53. DB2 の基本情報

資料名	資料番号	PDF ファイル名
「*IBM DB2 Universal Database コマンド・リファレンス*」	SC88-9140	db2n0j81
「*IBM DB2 Universal Database 用語集*」	資料番号なし	db2t0j81
「*IBM DB2 Universal Database メッセージ・リファレンス 第 1 巻*」	GC88-9152 (ハードコピーなし)	db2m1j81
「*IBM DB2 Universal Database メッセージ・リファレンス 第 2 巻*」	GC88-9153 (ハードコピーなし)	db2m2j81
「*IBM DB2 Universal Database 新機能*」	SC88-9158	db2q0j81

管理情報

これらの資料の情報は、DB2 データベース、データウェアハウス、およびフェデレーテッド・システムを効果的に設計し、インプリメントし、保守するために必要なトピックを扱っています。

表 54. 管理情報

資料名	資料番号	PDF ファイル名
「*IBM DB2 Universal Database 管理ガイド: プランニング*」	SC88-9135	db2d1j81
「*IBM DB2 Universal Database 管理ガイド: インプリメンテーション*」	SC88-9133	db2d2j81
「*IBM DB2 Universal Database 管理ガイド: パフォーマンス*」	SC88-9134	db2d3j81
「*IBM DB2 Universal Database 管理 API リファレンス*」	SC88-9136	db2b0j81
「*IBM DB2 Universal Database データ移動ユーティリティー ガイドおよびリファレンス*」	SC88-9142	db2dmj81
「*IBM DB2 Universal Database データ・リカバリーと高可用性 ガイドおよびリファレンス*」	SC88-9143	db2haj81
「*IBM DB2 Universal Database データウェアハウス・センター 管理ガイド*」	SC88-9165	db2ddj81
「*IBM DB2 Universal Database SQL リファレンス 第 1 巻*」	SC88-9155	db2s1j81
「*IBM DB2 Universal Database SQL リファレンス 第 2 巻*」	SC88-9156	db2s2j81

表 54. 管理情報 (続き)

資料名	資料番号	PDF ファイル名
「*IBM DB2 Universal Database システム・モニター ガイドおよびリファレンス*」	SC88-9157	db2f0j81

アプリケーション開発情報

これらの資料の情報は、 DB2 Universal Database (DB2 UDB) のアプリケーション開発者またはプログラマーが特に関心を持つ内容です。サポートされるさまざまなプログラミング・インターフェース (組み込み SQL、ODBC、JDBC、SQLJ、CLI など) を使用して DB2 UDB にアクセスするのに必要な資料とともに、サポートされる言語およびコンパイラーについても紹介されています。また、DB2 インフォメーション・センターをご使用の場合には、サンプル・プログラムのソース・コードの HTML バージョンにアクセスすることもできます。

表 55. アプリケーション開発情報

資料名	資料番号	PDF ファイル名
「*IBM DB2 Universal Database アプリケーション開発ガイド アプリケーションの構築および実行*」	SC88-9137	db2axj81
「*IBM DB2 Universal Database アプリケーション開発ガイド クライアント・アプリケーションのプログラミング*」	SC88-9138	db2a1j81
「*IBM DB2 Universal Database アプリケーション開発ガイド サーバー・アプリケーションのプログラミング*」	SC88-9139	db2a2j81
「*IBM DB2 Universal Database コール・レベル・インターフェース ガイドおよびリファレンス 第 1 巻*」	SC88-9159	db2l1j81
「*IBM DB2 Universal Database コール・レベル・インターフェース ガイドおよびリファレンス 第 2 巻*」	SC88-9160	db2l2j81
「*IBM DB2 Universal Database データウェアハウス・センター アプリケーション統合ガイド*」	SC88-9166	db2adj81
「*IBM DB2 Universal Database XML Extender 管理およびプログラミングのガイド*」	SC88-9172	db2sxj81

ビジネス・インテリジェンス情報

これらの資料の情報は、さまざまなコンポーネントを使用して、 DB2 Universal Database のデータウェアハウジング機能および分析機能を拡張する方法を説明しています。

表 56. ビジネス・インテリジェンス情報

資料名	資料番号	PDF ファイル名
「*IBM DB2 Warehouse Manager Standard Edition インフォメーション・カタログ・センター 管理ガイド*」	SC88-9167	db2dij81
「*IBM DB2 Warehouse Manager Standard Edition インストール・ガイド*」	GC88-9164	db2idj81
「*IBM DB2 Warehouse Manager Standard Edition DB2 Warehouse Manager を使用時の ETI ソリューション・コンバージョン・プログラムの管理*」	SC88-9894	iwhe1mstx80

DB2 Connect 情報

このカテゴリーの情報は、 DB2 Connect Enterprise Edition または DB2 Connect Personal Edition を使用して、メインフレーム・サーバーおよびミッドレンジ・サーバー上のデータにアクセスする方法を説明しています。

表 57. DB2 Connect 情報

資料名	資料番号	PDF ファイル名
「*IBM コネクティビティー 補足*」	資料番号なし	db2h1j81
「*IBM DB2 Connect Enterprise Edition 概説およびインストール*」	GC88-9145	db2c6j81
「*IBM DB2 Connect Personal Edition 概説およびインストール*」	GC88-9146	db2c1j81
「*IBM DB2 Connect ユーザーズ・ガイド*」	SC88-9147	db2c0j81

入門情報

このカテゴリーの情報は、サーバー、クライアント、および他の DB2 製品をインストールして構成する場合に役立ちます。

表 58. 入門情報

資料名	資料番号	PDF ファイル名
「*IBM DB2 Universal Database DB2 クライアント機能 概説およびインストール*」	GC88-9144 (ハードコピーなし)	db2itj81
「*IBM DB2 Universal Database DB2 サーバー機能 概説およびインストール*」	GC88-9148	db2isj81
「*IBM DB2 Universal Database DB2 Personal Edition 概説およびインストール*」	GC88-9150	db2i1j81
「*IBM DB2 Universal Database インストールおよび構成 補足*」	GC88-9149 (ハードコピーなし)	db2iyj81
「*IBM DB2 Universal Database DB2 Data Links Manager 概説およびインストール*」	GC88-9141	db2z6j81

チュートリアル情報

チュートリアル情報は、DB2 機能を紹介し、さまざまなタスクを実行する方法を示します。

表 59. チュートリアル情報

資料名	資料番号	PDF ファイル名
「*ビジネス・インテリジェンス・チュートリアル: データウェアハウス・センターの紹介*」	資料番号なし	db2tuj81
「*ビジネス・インテリジェンス・チュートリアル: データウェアハウジングの上級者向けガイド*」	資料番号なし	db2taj81
「*インフォメーション・カタログ・センター チュートリアル*」	資料番号なし	db2aij81
「*Video Central for e-business チュートリアル*」	資料番号なし	db2twj81
「*Visual Explain チュートリアル*」	資料番号なし	db2tvj81

オプショナル・コンポーネント情報

このカテゴリーの情報は、 DB2 のオプショナル・コンポーネントを使用する方法について説明しています。

表 60. オプショナル・コンポーネント情報

資料名	資料番号	PDF ファイル名
「*IBM DB2 Cube Views Guide and Reference*」	SC18–7298	db2aax81
「*IBM DB2 Query Patroller インストール、管理、使用法のガイド*」	GC88–9154	db2dwj81
「*IBM DB2 Spatial Extender and Geodetic Extender ユーザーズ・ガイドおよびリファレンス*」	SC88-9171	db2sbj81
「*IBM DB2 Universal Database Data Links Manager 管理ガイドおよびリファレンス*」	SC88-9169	db2z0x82
「*DB2 Net Search Extender 管理およびユーザーズ・ガイド*」 **注:** この資料の HTML 版は、HTML ドキュメンテーション CD からインストールされません。	SH88-8546	N/A

リリース・ノート

リリース・ノートは、ご使用の製品のリリースおよびフィックスパック・レベルに特有の追加情報を紹介します。また、リリース・ノートには、各リリース、アップデート、およびフィックスパックで組み込まれた資料上の更新の要約も含まれています。

表 61. リリース・ノート

資料名	資料番号	PDF ファイル名
「*DB2 リリース・ノート*」	「注」を参照。	「注」を参照。
「*DB2 インストール情報*」	製品 CD-ROM でのみ参照可能。	使用できません。

注: リリース・ノートは以下の形式で入手できます。

- XHTML およびテキスト形式 (製品 CD 内)
- PDF 形式 (PDF ドキュメンテーション CD 内)

さらに、リリース・ノートの中で、『*既知の問題と予備手段*』および『*リリース間の非互換性*』に関する部分は DB2 インフォメーション・センターにも表示されます。

UNIX ベースのプラットフォームでテキスト形式でリリース・ノートを確認するには、 `Release.Notes` ファイルを参照してください。このファイルは、`DB2DIR/Readme/%L` ディレクトリーに収録されています。 `%L` はロケール名を表しています。 `DB2DIR` は以下になります。

- AIX オペレーティング・システムの場合: /usr/opt/db2_08_01
- その他のすべての UNIX ベースのオペレーティング・システムの場合: /opt/IBM/db2/V8.1

関連概念:

- 451 ページの『DB2 資料とヘルプ』

関連タスク:

- 471 ページの『PDF ファイルからの DB2 資料の印刷方法』
- 472 ページの『DB2 の印刷資料の注文方法』
- 473 ページの『DB2 ツールからコンテキスト・ヘルプを呼び出す』

PDF ファイルからの DB2 資料の印刷方法

DB2 PDF ドキュメンテーション CD に収録されている DB2 資料を印刷することができます。 Adobe Acrobat Reader を使用すれば、資料全体または特定のページを印刷できます。

前提条件:

Adobe Acrobat Reader がインストールされていることを確認してください。 Adobe Acrobat Reader をインストールする必要がある場合、 Adobe Web サイト (www.adobe.com) から入手できます。

手順:

PDF ファイルから DB2 資料を印刷するには以下のようにします。

1. *DB2 PDF* ドキュメンテーション CD をドライブに挿入します。 UNIX オペレーティング・システムの場合、 DB2 PDF ドキュメンテーション CD をマウントします。 UNIX オペレーティング・システムで CD をマウントする方法については、「*概説およびインストール*」を参照してください。
2. index.htm を開きます。ブラウザー・ウィンドウにファイルが開きます。
3. 参照したい PDF のタイトルをクリックします。 Acrobat Reader で PDF が開きます。
4. **「ファイル」 → 「印刷」**を選択して、所要の資料の任意の部分を印刷します。

関連概念:

- 452 ページの『DB2 インフォメーション・センター』

関連タスク:

- 「*DB2 Universal Database サーバー機能 概説およびインストール*」の『CD-ROM のマウント (AIX)』
- 「*DB2 Universal Database サーバー機能 概説およびインストール*」の『HP-UX 上での CD-ROM のマウント』
- 「*DB2 Universal Database サーバー機能 概説およびインストール*」の『CD-ROM のマウント (Linux)』
- 472 ページの『DB2 の印刷資料の注文方法』

- 「*DB2 Universal Database サーバー機能 概説およびインストール*」の『CD-ROM のマウント (Solaris)』

関連資料:

- 465 ページの『DB2 PDF 資料および印刷された資料』

DB2 の印刷資料の注文方法

ハードコピー版の資料を望む場合には、以下のいずれかの方法で注文できます。

印刷資料の注文方法:

一部の国または地域では、印刷された資料を注文することもできます。お客様がお住まいの国または地域でこのサービスが利用可能かどうかを確認するには、お住まいの国または地域の IBM Publications Web サイトをご覧ください。資料のご注文が可能な場合、以下のようにすることができます。

- 正規の IBM 製品販売業者または営業担当員に連絡してください。お客様がお住まいの地域の IBM 担当員の情報については、お手数ですが IBM の Web サイト (www.ibm.com/planetwide) の IBM Worldwide Directory of Contacts で確認してください。
- IBM Publications Center (http://www.ibm.com/shop/publications/order) にアクセスしてください。なお、IBM Publications Center から資料を注文できない国もあります。

DB2 製品がご利用可能になった時点で、印刷された資料は *DB2 PDF* ドキュメンテーション *CD* にある PDF 形式の資料と同じものです。さらに、 *DB2* インフォメーション・センター *CD* に収録されている印刷された資料の内容もまた、これらと同じです。ただし、DB2 インフォメーション・センター CD には、 PDF 資料にない追加情報も含まれます (たとえば、SQL 管理作業や HTML サンプル)。 DB2 PDF ドキュメンテーション CD に収録されている資料の中には、ハードコピーとしてご注文できない資料もあります。

注: DB2 インフォメーション・センターは、 PDF またはハードコピー の資料よりも頻繁に更新されます。ドキュメンテーションの更新が入手可能になった時点でインストールするか、 DB2 インフォメーション・センター (http://publib.boulder.ibm.com/infocenter/db2help/) を参照して最新の情報を入手してください。

関連タスク:

- 471 ページの『PDF ファイルからの DB2 資料の印刷方法』

関連資料:

- 465 ページの『DB2 PDF 資料および印刷された資料』

DB2 ツールからコンテキスト・ヘルプを呼び出す

コンテキスト・ヘルプは、特定のウィンドウ、ノートブック、ウィザード、またはアドバイザーに関連したタスクまたはコントロールの情報を提供します。コンテキスト・ヘルプは、グラフィカル・ユーザー・インターフェースのある DB2 管理ツールおよび開発ツールから利用できます。コンテキスト・ヘルプには、以下の 2 種類があります。

- それぞれのウィンドウまたはノートブックにある**「ヘルプ」**ボタンからアクセス可能なヘルプ
- infopop (ポップアップ情報ウィンドウ)。これは、マウス・カーソルを特定のフィールドまたはコントロール上に置いたとき、またはウィンドウ、ノートブック、ウィザード、アドバイザー内でフィールドまたはコントロールを選択して F1 を押すと表示されます。

「ヘルプ」ボタンを押すと、概説、前提条件、およびタスク情報が表示されます。infopop は、それぞれのフィールドおよびコントロールについて説明します。

手順:

コンテキスト・ヘルプを呼び出すには、以下のようにします。

- ウィンドウおよびノートブックのヘルプを表示するには、いずれかの DB2 ツールを開始して、任意のウィンドウまたはノートブックを開きます。ウィンドウまたはノートブックの右下隅にある**「ヘルプ」**ボタンをクリックして、コンテキスト・ヘルプを呼び出します。

  また、それぞれの DB2 ツール・センターの上部にある**「ヘルプ」**メニュー項目からコンテキスト・ヘルプにアクセスすることもできます。

  ウィザードおよびアドバイザーでは、最初のページの「タスクの概要」リンクをクリックすると、コンテキスト・ヘルプを表示できます。
- ウィンドウまたはノートブック上の各コントロールの infopop ヘルプを表示するには、コントロールをクリックしてから、**F1** を押します。コントロールの詳細情報を示すポップアップ情報が、黄色いウィンドウに表示されます。

  注: フィールドまたはコントロールにマウス・カーソルを置いておくだけで infopops が表示されるようにするには、「ツール設定」ノートブックの**「文書 (Documentation)」**ページの**「infopops の自動表示」**チェック・ボックスを選択します。

  infopop に似た別のコンテキスト・ヘルプに、診断ポップアップ情報があります。これにはデータ入力規則が示されます。診断ポップアップ情報は、無効または不十分なデータが入力されたとき、紫色のウィンドウに表示されます。診断ポップアップ情報は、以下に関して表示されます。
  - 必須フィールド。
  - 日付フィールドのように、正確なフォーマットを必要とするデータのフィールド。

関連タスク:

- 462 ページの『DB2 インフォメーション・センターの呼び出し』
- 474 ページの『コマンド行プロセッサーからメッセージ・ヘルプを呼び出す』

- 474 ページの『コマンド行プロセッサーからコマンド・ヘルプを呼び出す』
- 475 ページの『コマンド行プロセッサーから SQL 状態ヘルプを呼び出す』
- 『DB2 インフォメーション・センターへのアクセス: Concepts help』
- 『DB2 UDB ヘルプの使用法: Common GUI help』
- 『DB2 インフォメーション・センターへのアクセスのロケーションの設定: Common GUI help』
- 『DB2 コンテキスト・ヘルプと資料へのアクセスを設定する: Common GUI help』

コマンド行プロセッサーからメッセージ・ヘルプを呼び出す

メッセージ・ヘルプは、メッセージが出された原因と、エラーへの応答として実行すべきアクションを説明します。

手順:

メッセージ・ヘルプを呼び出すには、コマンド行プロセッサーを開いて以下のように入力します。

```
? XXXnnnnn
```

ここで、*XXXnnnnn* は有効なメッセージ ID を表します。

たとえば、`? SQL30081` と入力すると、メッセージ SQL30081 に関するヘルプを表示します。

関連概念:

- 「メッセージ・リファレンス *第 1 巻*」の『メッセージの概要』

関連資料:

- 「コマンド・リファレンス」の『db2 - コマンド行プロセッサーの呼び出しコマンド』

コマンド行プロセッサーからコマンド・ヘルプを呼び出す

コマンド・ヘルプは、コマンド行プロセッサーでのコマンドの構文を説明します。

手順:

コマンド・ヘルプを呼び出すには、コマンド行プロセッサーを開いて以下のように入力します。

```
? command
```

ここで *command* はキーワードまたはコマンド全体を表します。

たとえば、`? catalog` と入力すると、すべての CATALOG コマンドに関するヘルプが表示され、 `? catalog database` と入力すると、 CATALOG DATABASE コマンドのヘルプだけが表示されます。

関連タスク:

- 473 ページの『DB2 ツールからコンテキスト・ヘルプを呼び出す』
- 462 ページの『DB2 インフォメーション・センターの呼び出し』
- 474 ページの『コマンド行プロセッサーからメッセージ・ヘルプを呼び出す』
- 475 ページの『コマンド行プロセッサーから SQL 状態ヘルプを呼び出す』

関連資料:

- 「コマンド・リファレンス」の『db2 - コマンド行プロセッサーの呼び出しコマンド』

コマンド行プロセッサーから SQL 状態ヘルプを呼び出す

DB2 Univerrsal Database は、 SQL ステートメントの結果の原因となったと考えられる条件の SQLSTATE 値を戻します。 SQLSTATE ヘルプは、SQL 状態および SQL 状態クラス・コードの意味を説明します。

手順:

SQL 状態ヘルプを呼び出すには、コマンド行プロセッサーを開いて以下のように入力します。

```
? sqlstate または ? class code
```

ここで、*sqlstate* は有効な 5 桁の SQL 状態を、*class code* は SQL 状態の最初の 2 桁を表します。

たとえば、`? 08003` を指定すると SQL 状態 08003 のヘルプが表示され、 `? 08` を指定するとクラス・コード 08 のヘルプが表示されます。

関連タスク:

- 462 ページの『DB2 インフォメーション・センターの呼び出し』
- 474 ページの『コマンド行プロセッサーからメッセージ・ヘルプを呼び出す』
- 474 ページの『コマンド行プロセッサーからコマンド・ヘルプを呼び出す』

DB2 チュートリアル

DB2® チュートリアルは、 DB2 Universal Database のさまざまな機能について学習するのを支援します。このチュートリアルでは、アプリケーションの開発、SQL 照会のパフォーマンス調整、データウェアハウスの処理、メタデータの管理、および DB2 を使用した Web サービスの開発の各分野で、段階的なレッスンが用意されています。

はじめに:

インフォメーション・センター (http://publib.boulder.ibm.com/infocenter/db2help/) から、このチュートリアルの XHTML 版を表示できます。

チュートリアルの中で、サンプル・データまたはサンプル・コードを使用する場合があります。個々のタスクの前提条件については、それぞれのチュートリアルを参照してください。

DB2 Universal Database チュートリアル:

以下に示すチュートリアルのタイトルをクリックすると、そのチュートリアルを表示できます。

ビジネス・インテリジェンス・チュートリアル: データウェアハウス・センターの紹介
データウェアハウス・センターを使用して簡単なデータウェアハウジング・タスクを実行します。

ビジネス・インテリジェンス・チュートリアル: データウェアハウジングの上級者向けガイド
データウェアハウス・センターを使用して高度なデータウェアハウジング・タスクを実行します。

インフォメーション・カタログ・センター・チュートリアル
インフォメーション・カタログを作成および管理して、インフォメーション・カタログ・センターを使用してメタデータを配置し使用します。

Visual Explain チュートリアル
Visual Explain を使用して、パフォーマンスを向上させるために SQL ステートメントを分析し、最適化し、調整します。

DB2 トラブルシューティング情報

DB2® 製品を使用する際に役立つ、トラブルシューティングおよび問題判別に関する広範囲な情報を利用できます。

DB2 ドキュメンテーション
トラブルシューティング情報は、DB2 インフォメーション・センター、および DB2 ライブラリーに含まれる PDF 資料の中でご利用いただけます。DB2 インフォメーション・センターで、 (ブラウザー・ウィンドウの左側の) ナビゲーション・ツリーの「サポートおよびトラブルシューティング (Support and troubleshooting)」ブランチを参照すると、 DB2 トラブルシューティング・ドキュメンテーションの詳細なリストが見つかります。

DB2 Technical Support の Web サイト
現在問題が発生していて、考えられる原因とソリューションを検索したい場合は、 DB2 Technical Support の Web サイトを参照してください。Technical Support サイトには、最新の DB2 出版物、TechNotes、プログラム診断依頼書 (APAR)、フィックスパック、DB2 内部エラー・コードの最新リスト、その他のリソースが用意されています。この知識ベースを活用して、問題に対する有効なソリューションを探し出すことができます。

DB2 Technical Support の Web サイト (http://www.ibm.com/software/data/db2/udb/winos2unix/support) にアクセスしてください。

DB2 Problem Determination Tutorial Series
DB2 製品で作業中に直面するかもしれない問題を素早く識別し、解決する方法に関する情報を見つけるには、 DB2 Problem Determination Tutorial Series の Web サイトを参照してください。あるチュートリアルでは、使用可能な DB2 問題判別機能およびツールを紹介し、それらをいつ使用すべきかを判断する助けを与えます。別のチュートリアルは、『データベース・エ

ンジン問題判別 (Database Engine Problem Determination)』、『パフォーマンス問題判別 (Performance Problem Determination)』、『アプリケーション問題判別 (Application Problem Determination)』などの関連トピックを扱っています。

DB2 Technical Support (http://www.ibm.com/software/data/support/pdm/db2tutorials.html) には、 DB2 問題判別チュートリアルがすべて揃っています。

関連概念:

- 452 ページの『DB2 インフォメーション・センター』
- 「*問題判別の手引き*」の『Introduction to Problem Determination - DB2 テクニカル・サポートのチュートリアル』

アクセス支援

アクセス支援機能は、身体に障害のある (身体動作が制限されている、視力が弱いなど) ユーザーがソフトウェア製品を十分活用できるように支援します。 DB2® バージョン 8 製品に備わっている主なアクセス支援機能は、以下のとおりです。

- すべての DB2 機能は、マウスの代わりにキーボードを使ってナビゲーションできます。詳細については、『キーボードによる入力およびナビゲーション』を参照してください。
- DB2 ユーザー・インターフェースのフォント・サイズおよび色をカスタマイズすることができます。詳細については、 478 ページの『アクセスしやすい表示』を参照してください。
- DB2 製品は、Java™ Accessibility API を使用するアクセス支援アプリケーションをサポートします。詳細については、 478 ページの『支援テクノロジーとの互換性』を参照してください。
- DB2 資料は、アクセスしやすい形式で提供されています。詳細については、 478 ページの『アクセスしやすい資料』を参照してください。

キーボードによる入力およびナビゲーション

キーボード入力

キーボードだけを使用して DB2 ツールを操作できます。マウスを使って実行できる操作は、キーまたはキーの組み合わせによっても実行できます。標準のオペレーティング・システム・キー・ストロークを使用して、標準のオペレーティング・システム操作を実行できます。

キーまたはキーの組み合わせによって操作を実行する方法について、詳しくは キーボード・ショートカットおよびアクセラレーター: Common GUI help を参照してください。

キーボード・ナビゲーション

キーまたはキーの組み合わせを使用して、 DB2 ツールのユーザー・インターフェースをナビゲートできます。

キーまたはキーの組み合わせによって DB2 ツールをナビゲートする方法の詳細については、キーボード・ショートカットおよびアクセラレーター: Common GUI help を参照してください。

キーボード・フォーカス

UNIX® オペレーティング・システムでは、アクティブ・ウィンドウの中で、キー・ストロークによって操作できる領域が強調表示されます。

アクセスしやすい表示

DB2 ツールには、視力の弱いユーザー、その他の視力障害をもつユーザーのためにアクセシビリティーを向上させる機能が備わっています。これらのアクセシビリティー拡張機能には、フォント・プロパティーのカスタマイズを可能にする機能も含まれています。

フォントの設定

「ツール設定」ノートブックを使用して、メニューおよびダイアログ・ウィンドウに使用されるテキストの色、サイズ、およびフォントを選択できます。

フォント設定に関する詳細情報は、メニューおよびテキストのフォントを変更する: Common GUI help を参照してください。

色に依存しない

本製品のすべての機能を使用するために、ユーザーは必ずしも色を識別する必要はありません。

支援テクノロジーとの互換性

DB2 ツールのインターフェースは、Java Accessibility API をサポートします。これによって、スクリーン・リーダーその他の支援テクノロジーを DB2 製品で利用できるようになります。

アクセスしやすい資料

DB2 形式は、ほとんどの Web ブラウザーで表示可能な XHTML 1.0 形式で提供されています。 XHTML により、ご使用のブラウザーに設定されている表示設定に従って資料を表示できます。さらに、スクリーン・リーダーや他の支援テクノロジーを使用することもできます。

シンタックス・ダイアグラムはドット 10 進形式で提供されます。この形式は、スクリーン・リーダーを使用してオンライン・ドキュメンテーションにアクセスする場合にのみ使用できます。

関連概念:

- 479 ページの『ドット 10 進シンタックス・ダイアグラム』

関連タスク:

- 『キーボード・ショートカットおよびアクセラレーター: Common GUI help』
- 『メニューおよびテキストのフォントを変更する: Common GUI help』

ドット 10 進シンタックス・ダイアグラム

スクリーン・リーダーを使用してインフォメーション・センターを利用するユーザーのために、シンタックス・ダイアグラムがドット 10 進形式で提供されます。

ドット 10 進形式では、各シンタックス・エレメントは別々の行に書き込まれます。複数のシンタックス・エレメントが常に同時に存在する (または常に同時に不在の) 場合、単一のコンパウンド・シンタックス・エレメントとみなせるので同一行に表示できます。

各行は、ドット 10 進数で開始します。たとえば、3 または 3.1 ないしは 3.1.1 です。こうした数を適切に聞き取るには、スクリーン・リーダーが句読点を読み取るように設定されていることを確認してください。同じドット 10 進数を持つすべてのシンタックス・エレメント (たとえば、 3.1 という数値を持つすべてのシンタックス・エレメント) は、相互に排他的な代替エレメントです。 3.1 USERID および 3.1 SYSTEMID という行を聞き取る場合、シンタックスには両方ではなく USERID または SYSTEMID のどちらかが含まれることが分かります。

ドット 10 進レベルは、ネストのレベルを表示します。たとえば、ドット 10 進数 3 のシンタックス・エレメントの後に、一連のドット 10 進数 3.1 のシンタックス・エレメントが続きます。 3.1 の番号が付されたシンタックス・エレメントすべては、番号 3 の付されたシンタックス・エレメントに従属します。

シンタックス・エレメントに関する情報を追加するため、ドット 10 進数の次に特定のワードおよびシンボルが使用されます。 時折、こうしたワードおよびシンボルはエレメントの最初に表示される場合もあります。 簡単に識別するため、ワードやシンボルがシンタックス・エレメントの一部である場合には、円記号 (¥) 文字が先頭に付きます。 * シンボルはドット 10 進数の次に使用でき、シンタックス・エレメントが反復することを示します。 たとえば、ドット 10 進数 3 のシンタックス・エレメント *FILE は、3 ¥* FILE という形式になります。 3* FILE という形式は、シンタックス・エレメント FILE が反復されることを示します。 3* ¥* FILE という形式は、シンタックス・エレメント * FILE が反復されることを示します。

シンタックス・エレメントのストリングを分離するのに使用されるコンマなどの文字は、シンタックス内の分離する項目の直前に表示されます。 こうした文字は、それぞれの項目と同一行に表示するか、同じドット 10 進数を持つ関連する項目のある別の行に表示できます。 またその行には、シンタックス・エレメントに関する情報を提供する別のシンボルを表示することも可能です。 たとえば、複数の LASTRUN および DELETE シンタックス・エレメントを使用している場合には、5.1*、5.1 LASTRUN、および 5.1 DELETE という行は、エレメントをコンマで区切る必要があります。区切り文字が指定されないと、各シンタックス・エレメントを区切るのにブランクが使用されると想定されます。

シンタックス・エレメントの前に % シンボルが付く場合、他の箇所で定義されている参照であることを示します。 % シンボルの後のストリングは、リテラルではなくシンタックス・フラグメントの名前です。 たとえば、2.1 %OP1 という行は別のシンタックス・フラグメント OP1 を参照すべきことを意味します。

以下のワードおよびシンボルが、ドット 10 進数の次に使用されます。

- ? は、オプションのシンタックス・エレメントであることを表します。? シンボルが後に続くドット 10 進数は、対応するドット 10 進数のシンタックス・エレメント、および任意の従属のシンタックス・エレメントがオプションであることを示します。ドット 10 進数の付いたシンタックス・エレメントが 1 つしかない場合、 ? シンボルはそのシンタックス・エレメントと同じ行に表示されます (たとえば、5? NOTIFY)。ドット 10 進数の付いたシンタックス・エレメントが複数ある場合、 ? シンボルだけで行に表示され、その後にオプションのシンタックス・エレメントが続きます。たとえば、「5 ?、5 NOTIFY、および 5 UPDATE」という行を聞き取る場合、シンタックス・エレメント NOTIFY および UPDATE がオプションである、つまりそのいずれかを選択でき、どちらも選択しないこともできることが分かります。 ? シンボルは、線路型ダイアグラムのバイパス線に相当します。
- ! は、デフォルトのシンタックス・エレメントであることを表します。! シンボルおよびシンタックス・エレメントが後に続くドット 10 進数は、そのシンタックス・エレメントが、同じドット 10 進数を共用するシンタックス・エレメントすべてのデフォルト・オプションであることを示します。同じドット 10 進数を共用するシンタックス・エレメントのうち 1 つだけに、 ! シンボルを指定できます。たとえば、「2? FILE、2.1! (KEEP)、および 2.1 (DELETE)」という行を聞き取る場合、 FILE キーワードのデフォルト・オプションは (KEEP) になります。この例では、FILE キーワードを含めてもオプションを指定しない場合には、デフォルト・オプション KEEP が適用されます。デフォルト・オプションは、次に高位のドット 10 進数にも適用されます。 この例の場合、FILE キーワードが省略されると、デフォルトの FILE(KEEP) が使用されます。しかし、「2? FILE、2.1、2.1.1! (KEEP)、および 2.1.1 (DELETE)」という行を聞き取る場合、デフォルト・オプション KEEP は次に高位のドット 10 進数 2.1 (関連キーワードを持っていない) にのみ適用され、 2? FILE には適用されません。キーワード FILE が省略されると、どれも使用されません。
- * は、0 回以上反復できるシンタックス・エレメントを示します。 * シンボルが後に続くドット 10 進数は、このシンタックス・エレメントが 0 回以上使用できること、つまりオプションであり、なおかつ反復できることを表します。たとえば、5.1* データ域という行を聞き取る場合、 1 つまたは複数のデータ域を含めるか、またはデータ域を全く含めないことが可能です。「3*、3 HOST、および 3 STATE」という行を聞き取る場合、 HOST、STATE をどちらか一方または両方同時に含めるか、どちらも含めないことができます。

  注:

  1. ドット 10 進数の後にアスタリスク (*) が付き、ドット 10 進数の付いた項目が 1 つしかない場合には、同じ項目を複数回反復できます。
  2. ドット 10 進数の後にアスタリスクが付き、ドット 10 進数の付いた項目が複数ある場合、リストから複数の項目を使用できますが、各項目を複数回使用することはできません。 前述の例では、HOST STATE と書くことはできますが、HOST HOST とは書けません。
  3. * シンボルは、線路型シンタックス・ダイアグラムのループバック線に相当します。
- + は、1 回以上含める必要のあるシンタックス・エレメントであることを示します。 + シンボルが後に続くドット 10 進数は、このシンタックス・エレメントを 1 回以上含める必要があること、つまり少なくとも 1 回は含める必要があり、反

復できることを表します。たとえば、「6.1+ データ域」という行を聞き取る場合、データ域を少なくとも 1 回は含めなければなりません。「2+、2 HOST、および 2 STATE」という行を聞き取る場合には、 HOST、STATE、またはその両方を含める必要があります。 * シンボルと同様に、+ シンボルは、ドット 10 進数の付いた項目が 1 つしかない場合に限り、その特定の項目のみを反復できます。 * シンボルと同様、 + シンボルは線路型シンタックス・ダイアグラムのループバック線に相当します。

関連概念:

- 477 ページの『アクセス支援』

関連タスク:

- 『キーボード・ショートカットおよびアクセラレーター: Common GUI help』

関連資料:

- 「*SQL リファレンス 第 2 巻*」の『構文図の見方』

DB2 Universal Database 製品の共通基準認証

DB2 Universal Database は、 Common Criteria の評価検定レベル 4 (EAL4) で認証の評価を受けています。 Common Criteria の詳細については、以下の Common Criteria の Web サイトを参照してください。 http://niap.nist.gov/cc-scheme/

付録 L. 特記事項

本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒106-0032
東京都港区六本木 3-2-31
IBM World Trade Asia Corporation
Licensing

以下の保証は、国または地域の法律に沿わない場合は、適用されません。 IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。 IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム(本プログラムを含む)との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM Canada Limited
Office of the Lab Director
8200 Warden Avenue
Markham, Ontario
L6G 1C7
CANADA

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、 IBM より提供されます。

この文書に含まれるいかなるパフォーマンス・データも、管理環境下で決定されたものです。そのため、他の操作環境で得られた結果は、異なる可能性があります。一部の測定が、開発レベルのシステムで行われた可能性がありますが、その測定値が、一般に利用可能なシステムのものと同じである保証はありません。さらに、一部の測定値が、推定値である可能性があります。実際の結果は、異なる可能性があります。お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。 IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。 IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

IBM の将来の方向または意向に関する記述については、予告なしに変更または撤回される場合があり、単に目標を示しているものです。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

著作権使用許諾:

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。

それぞれの複製物、サンプル・プログラムのいかなる部分、またはすべての派生した創作物には、次のように、著作権表示を入れていただく必要があります。

商標

以下は、IBM Corporation の商標です。

ACF/VTAM
AISPO
AIX
AIXwindows
AnyNet
APPN
AS/400
BookManager
C Set++
C/370
CICS
Database 2
DataHub
DataJoiner
DataPropagator
DataRefresher
DB2
DB2 Connect
DB2 Extenders
DB2 OLAP Server
DB2 Information Integrator
DB2 Query Patroller
DB2 Universal Database
Distributed Relational Database Architecture
DRDA
eServer
Extended Services
FFST
First Failure Support Technology
IBM
IMS
IMS/ESA
iSeries
LAN Distance
MVS
MVS/ESA
MVS/XA
Net.Data
NetView
OS/390
OS/400
PowerPC
pSeries
QBIC
QMF
RACF
RISC System/6000
RS/6000
S/370
SP
SQL/400
SQL/DS
System/370
System/390
SystemView
Tivoli
VisualAge
VM/ESA
VSE/ESA
VTAM
WebExplorer
WebSphere
WIN-OS/2
z/OS
zSeries

以下は、それぞれ各社の商標または登録商標です。

Microsoft、Windows、Windows NT および Windows ロゴは、Microsoft Corporation の米国およびその他の国における商標です。

Pentium は、Intel Corporation の米国およびその他の国における商標です。

Java およびすべての Java 関連の商標およびロゴは、Sun Microsystems, Inc. の米国およびその他の国における商標または登録商標です。

UNIX は、The Open Group の米国およびその他の国における登録商標です。

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。

索引

日本語, 数字, 英字, 特殊文字の順に配列されています。なお, 濁音と半濁音は清音と同等に扱われています。

[ア行]

[カ行]

[サ行]

[タ行]

[ナ行]

[ハ行]

[マ行]

[ヤ行]

[ラ行]

[ワ行]

A

B

C

D

E

F

G

I

J

K

L

M

N

IBM と連絡をとる

技術上の問題がある場合は、お客様サポートにご連絡ください。

製品情報

DB2 Universal Database 製品に関する情報は、
http://www.ibm.com/software/data/db2/udb から入手できます。

このサイトには、技術ライブラリー、資料の注文方法、製品のダウンロード、ニュースグループ、フィックスパック、ニュース、および Web リソースへのリンクに関する最新情報が掲載されています。

米国以外の国で IBM に連絡する方法については、 IBM Worldwide ページ (www.ibm.com/planetwide) にアクセスしてください。

Printed in Japan

SC88-9133-01

日本アイ・ビー・エム株式会社
〒106-8711　東京都港区六本木3-2-12